1/5/1
DIALOG(R)File 351:Derwent WPI

011170213 **Image available**
WPI Acc No: 1997-148138/199714
XRPX Acc No: N97-122560

User-level process scheduler for computer system - has scheduler that performs scheduling of group of other user processes, which have priorities lower than first priority, and causes such user processes to operate or run

Patent Assignee: FUJITSU LTD (FUIT)
Inventor: KAMADA J; ONO E; YUHARA M

Number of Countries: 003 Number of Patents: 005
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
GB 2304211	A	19970312	GB 967228	A	19960404	199714	B
JP 9054699	A	19970225	JP 95205424	A	19950811	199718	
GB 2304211	B	20000419	GB 967228	A	19960404	200022	
US 6108683	A	20000822	US 96621181	A	19960321	200042	
US 6668269	B1	20031223	US 96621181	A	19960321	200408	
			US 99474100	A	19991229		

Priority Applications (No Type Date): JP 95205424 A 19950811
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
GB 2304211	A		106	G06F-009/46	
JP 9054699	A		29	G06F-009/46	
GB 2304211	B			G06F-009/46	
US 6108683	A			G06F-009/46	
US 6668269	B1			G06F-009/00	Div ex application US 96621181 Div ex patent US 6108683

Abstract (Basic): GB 2304211 A

The system includes a scheduler that belongs to a user process and allocates a central processing unit (CPU) to executable ones of the processes in descending order of their priorities. The scheduler is provided in a fixed-priority process scheduler space, namely in a real-time class process scheduler space, and is given the top level of priority that would generally be assigned to a class of programs requiring real time operation. The scheduler performs the scheduling of a group of other user processes, which have priorities lower than the first priority, and causes such user processes to operate or run.

ADVANTAGE - Ensures utilisation of CPU in user process required bu multi-media system while scheduling is carried by user program.

Dwg.6/39

Title Terms: USER; LEVEL; PROCESS; COMPUTER; SYSTEM; PERFORMANCE; SCHEDULE; GROUP; USER; PROCESS; PRIORITY; LOWER; FIRST; PRIORITY; CAUSE; USER; PROCESS; OPERATE; RUN
Derwent Class: T01
International Patent Class (Main): G06F-009/00; G06F-009/46
File Segment: EPI

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－54699

(43)公開日　平成９年(1997)２月25日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/46	３４０		Ｇ０６Ｆ　9/46	３４０Ｂ	

審査請求　未請求　請求項の数31　ＯＬ　（全 29 頁）

(21)出願番号　特願平7－205424

(22)出願日　平成７年(1995)８月11日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　蒲田　順
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(72)発明者　湯原　雅信
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(72)発明者　小野　越夫
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

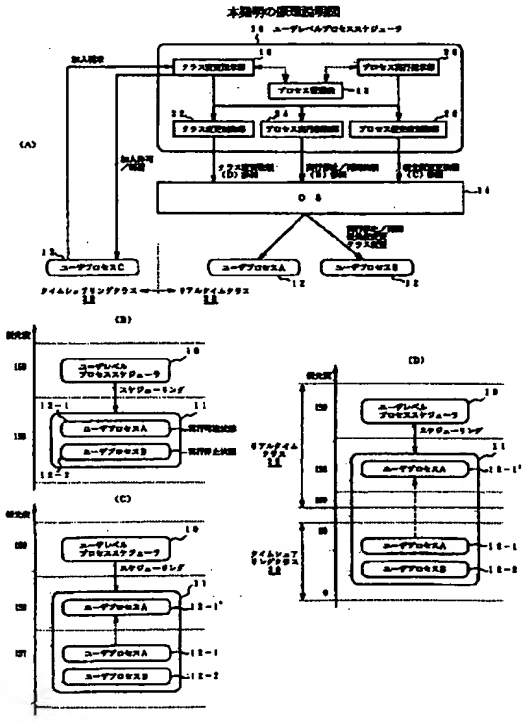
(74)代理人　弁理士　竹内　進　（外１名）

(54)【発明の名称】　計算機のプロセススケジューラ

(57)【要約】

【課題】ＯＳがサポートするスケジューラを利用してユーザプログラムでスケジューリングを行いながらも、マルチメディアシステムが必要としているユーザプロセスでのＣＰＵ利用を保証する。

【解決手段】ＯＳ１４によりサポートされ、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定めると共に、優先度はユーザプロセスからの指定で変更可能であり、実行可能なプロセスのうち優先度の高いプロセスからＣＰＵを割当てて動作させる計算機のプロセススケジューラを利用する。この固定優先度プロセススケジューラ、即ちリアルタイムクラス２８のプロセススケジューラ空間に、ユーザレベルプロセススケジューラ１０を設ける。ユーザレベルプロセススケジューラ１０は、リアルタイムクラス２８の第１優先度１５９を持ち、第１優先度１５９より低い優先度を持つ他のユーザプロセス群１１をスケジューリングして動作させる。即ち、ＣＰＵ割当てを決めてユーザプロセスの動作をＯＳ１４に依頼する。



1

【特許請求の範囲】

【請求項１】オペレーティングシステムによりサポートされ、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定めると共に、該優先度はユーザプロセスからの指定で変更可能であり、実行可能な複数のプロセスのうち優先度の高いプロセスからＣＰＵを割り当てて動作させる計算機のプロセススケジューラに於いて、

ユーザプロセスに属すると共に前記固定優先度スケジューリングの第１優先度を持ち、前記第１優先度より低い優先度を持つ他のユーザプロセス群をスケジューリングして動作させる（ＣＰＵ割当てを決めてユーザプロセスの動作をＯＳに依頼する）ユーザレベルプロセススケジューラを設けたことを特徴とする計算機のプロセススケジューラ。

【請求項２】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、第１優先度をもつ複数のユーザプロセス毎にリンクされ、各ユーザレベルプロセススケジューラは自己がリンクしたユーザプロセスをスケジューリングして動作させることを特徴とする計算機のプロセススケジューラ。

【請求項３】請求項１又は２記載の計算機のプロセススケジューラに於いて、前記第１優先度のユーザレベルプロセススケジューラは、自己がスケジュールしているユーザプロセス群中に、第２優先度以下のユーザレベルプロセススケジューラを設け、前記上位から下位に階層的に他のユーザプロセス群をスケジューリングして動作させることを特徴とする計算機のプロセススケジューラ。

【請求項４】請求項１記載の計算機のプロセススケジューラに於いて、第一優先度をもつユーザレベルスケジューラを複数設け、各ユーザレベルプロセススケジューラが個別に自己のユーザプロセス群をスケジューリングして実行することを特徴とする計算機のプロセススケジューラ。

【請求項５】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスからの要求に基づくクラス変更指示部からの指示、又はプロセス管理表の参照によるプロセス実行指示部からの指示に従い、指定されたユーザプロセスの実行、停止又は再開をオペレーティングシステムに依頼するプロセス実行制御部を備えたことを特徴とする計算機のプロセススケジューラ。

【請求項６】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスからの要求に基づくクラス変更指示部からの指示、又はプロセス管理表の参照によるプロセス実行指示部からの指示に従い、指定されたユーザプロセスの優先度を第１優先度に変更してユーザプロセスを実行させ、下位の優先度に変更した他のユーザプロセスを停止させることで、ユーザプロセスの実行、停止、又は再開をオペレーティングシステムに依頼するプロセス優先度制御部を備えたことを特徴とする計算機のプロセススケジューラ。

2

【請求項７】請求項１記載の計算機のプロセススケジューラに於いて、

前記オペレーティングシステムは、高い優先度の領域にある固定優先度のリアルタイムクラスに対し、低い優先度の領域にある優先度をプロセス状態に応じて動的に変化させるタイムシェアリングクラスを有する場合、前記ユーザレベルプロセススケジューラは、ユーザプロセスからの要求に基づくクラス変更指示部からの指示、又はプロセス管理表の参照によるプロセス実行指示部からの指示に従い、指定されたクラスの変更をオペレーティングシステムに依頼して、ユーザプロセスの実行、停止又は再開させるクラス変更制御部を備えたことを特徴とする計算機のプロセススケジューラ。

【請求項８】請求項５乃至７記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスのＣＰＵ使用時間の指定を伴なう加入要求に対し、前記プロセス管理表を参照して現在加入状態にあるユーザプロセス群の要求ＣＰＵ時間の合計時間を求め、該合計時間が規定時間以下であれば前記ユーザプロセスをの加入を許可し、規定値を超えていれば加入を拒否することを特徴とする計算機のプロセススケジューラ。

【請求項９】請求項８記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの加入を許可する場合、前記プロセス実行制御部に加入要求プロセスの実行停止状態への変更を指示し、且つ前記クラス変更制御部に、低い優先度の領域にある動的優先度のタイムシェアリングクラスから高い優先度の領域にある固定優先度のリアルタイムクラスへの変更を指示し、前記プロセス優先度制御部に第２優先度への変更を指示し、更にユーザプロセスをプロセス管理表に登録した後に、加入許可を通知することを特徴とする計算機のプロセススケジューラ。

【請求項１０】請求項８記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの加入を許可する場合、前記クラス変更制御部にリアルタイムクラスへの変更を指示し、且つ前記プロセス優先度制御部にリアルタイムクラスの第１優先度より下位の優先度への変更を指示し、更にユーザプロセスをプロセス管理表に登録した後に、加入許可を通知することを特徴とする計算機のプロセススケジューラ。

【請求項１１】請求項８記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの加入を許可する場合、前記クラス変更制御部にタイムシェアリングクラスへの変更を指示し、更にユーザプロセスをプロセス管理表に登録した後に、加入許可を通知することを特徴とする計算機のプロセススケジューラ。

【請求項１２】請求項５乃至７記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの脱会要求に対し、前記プロセス管理表からユーザプロセスを削除すると共に、前記クラス変更制御部に脱会要求を行ったユーザプロセスのタイムシェアリングクラスへの変更を指示することを特徴とする計算機のプロセススケジューラ。

【請求項１３】請求項５乃至７記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスからの通知により新規なユーザプロセスの加入と脱会を処理することを特徴とする計算機のプロセススケジューラ。

【請求項１４】請求項５乃至７記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、オペレーティングシステムからの通知により新規なユーザプロセスの加入と脱会を処理することを特徴とする計算機のプロセススケジューラ。

【請求項１５】請求項５乃至７記載の計算機のプロセススケジューラに於いて、前記プロセス実行制御部、クラス変更制御部、プロセス優先度制御部に対する変更要求の情報を、アプリケーションにリンクされたライブラリを経由して、アプリケーションから要求指示を受け、オペレーティングシステムに変更を依頼することを特徴とする計算機のプロセススケジューラ。

【請求項１６】請求項５乃至７記載の計算機のプロセススケジューラに於いて、前記プロセス実行制御部、クラス変更制御部、プロセス優先度制御部に対する変更要求のスケジューリング仕様をユーザプロセスの１つで生成し、該ユーザプロセスのスケジューリング仕様に基づき前記ユーザレベルプロセススケジューラに変更要求を通知して前記オペレーティングシステムに変更を依頼することを特徴とする計算機のプロセススケジューラ。

【請求項１７】請求項１記載の計算機のプロセススケジューラにおいて、前記ユーザレベルプロセススケジューラのスケジューリング対象となるユーザプロセスは、複数のスレッドから構成され、該ユーザプロセスの一部であるスレッドスケジューラにより前記複数のスレッドをスケジューリングすることを特徴とする計算機のプロセススケジューラ。

【請求項１８】請求項１記載の計算機のプロセススケジューラに於いて、ユーザプロセスの前記ユーザレベルプロセススケジューラは、プロセス管理表に登録された１又は複数のユーザプロセスにＣＰＵ時間を割り当てる際、ＣＰＵ割当時間帯にユーザプロセスが１つしか存在しないように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステムに依頼することを特徴とする計算機のプロセススケジューラ。

【請求項１９】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、プロセス管理表に登録された１又は複数のユーザ

プロセスにＣＰＵ時間を割り当てる際、ＣＰＵ割当時間帯の中に複数のユーザプロセスが時分割で存在するように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステムに依頼することを特徴とする計算機のプロセススケジューラ。

【請求項２０】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定期間内に複数のユーザプロセスと自分自身を含めてＣＰＵ時間を順番に割り当てることにより、複数のユーザプロセスの動作を一定期間内で保証したことを特徴とする計算機のプロセススケジューラ。

【請求項２１】請求項１記載の計算機のプロセススケジューラに於いて、前記プロセススケジューラは、一定の周期を設定し、各周期内で複数のユーザプロセスと自分自身を含めたＣＰＵ時間を順番に割り当てることを繰り返すことにより、複数のユーザプロセスで動作を保証したことを特徴とする計算機のプロセススケジューラ。

【請求項２２】請求項２１記載の計算機のプロセススケジューラに於いて、複数のユーザプロセスが持つ動作周期が異なる場合、前記ユーザレベルプロセススケジューラは、異なる動作周期の組合わせで１つの周期を決定し、該周期内で複数のユーザプロセスに使用可能なＣＰＵ時間を順番に割り当てることにより、複数のユーザプロセスの動作を保証したことを特徴とする計算機のプロセススケジューラ。

【請求項２３】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定時間又は一定周期において複数のユーザプロセスに割り当てたＣＰＵ時間の占める割合が規定値以内の時、他のユーザプロセスに対するＣＰＵ時間の割当てを許容し、前記規定値を越えた時、他のユーザプロセスに対するＣＰＵ時間の割当てを拒否することを特徴とする計算機のプロセススケジューラ。

【請求項２４】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定期間又は一定周期でユーザプロセスにＣＰＵ時間を割り当てる場合、連続したＣＰＵ時間を割り当てることを特徴とする計算機のプロセススケジューラ。

【請求項２５】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定期間内又は一定周期内で複数のユーザプロセスのＣＰＵ時間を割り当てる場合、複数のユーザプロセスのＣＰＵ時間の各々を分割して割り当てることを特徴とする計算機のプロセススケジューラ。

【請求項２６】請求項１記載の計算機のプロセススケジューラに於いて、プロセス管理表に登録されたユーザプロセスとして、高い優先度の領域にある固定優先度のリアルタイムクラスに属するユーザプロセスと、低い優先度の領域にあり優先度を動的に変化させるタイムシェアリングクラスに属するユーザプロセスが存在する場合、

前記ユーザレベルプロセススケジューラは、前記リアルタイムクラスのユーザプロセスに対するＣＰＵ時間の割り当てを制限してタイムシェアリングクラスのユーザプロセスにＣＰＵ時間を割当てることを特徴とする計算機のプロセススケジューラ。

【請求項２７】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスの実行停止を検出して他のユーザプロセスを動作可能とすることを特徴とする計算機のプロセススケジューラ。

【請求項２８】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスにＣＰＵ時間を割り当てる際に、該ユーザプロセスに加え同じ優先度で実行可能なブロック検出プロセスを設け、前記ユーザプロセスがブロックした際に前記ブロック検出プロセスにＣＰＵ時間を割当ててブロックを検出して通知させ、該通知により前記ユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とすることを特徴とする計算機のプロセススケジューラ。

【請求項２９】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスにＣＰＵ時間を割り当てる際に、該ユーザプロセスに加え低い優先度で実行可能なブロック検出プロセスを設け、前記ユーザプロセスがブロックした際に前記ブロック検出プロセスにＣＰＵ時間を割り当ててブロックを検出して通知させ、該通知により前記ユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とすることを特徴とする計算機のプロセススケジューラ。

【請求項３０】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ブロック中のユーザプロセスが実行可能な状態になったことを検出して動作させることを特徴とする計算機のプロセススケジューラ。

【請求項３１】請求項１記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、オペレーティングシステムによるユーザプロセスのブロックまたはブロック回復による実行再開の通知を検出して他のユーザプロセスの実行又は停止を行うことを特徴とする計算機のプロセススケジューラ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、オペレーティングシステムでサポートされたプロセススケジューラを使用してユーザレベルのプロセススケジューラを機能させる計算機のプロセススケジューラに関し、特に、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定め、実行可能なプロセスのうち優先度の高いプロセスからＣＰＵを割り当てて動作させるプロセススケジューラを使用してユーザレベルプロセススケジューラを機能させる計算機のプロセススケジューラに関する。

【０００２】

【従来の技術】計算機では、一般に複数のプログラムを平行して動作させるため、プロセスと呼ばれる実行単位が同時に複数存在できるようにしている。この時、アルゴリズムにしたがって各プロセスにＣＰＵを割り当てる必要がある。このプロセスに対するＣＰＵの割当てをプロセススケジューリングという。

【０００３】従来のプロセススケジューリングは、バッチシステムあるいはタイムシェアリングシステムであり、システム全体でのＣＰＵの利用効率を高めることを目標として作られてきた。タイムシェアリングシステムでは、ユーザに対する応答性能を短くすることも考慮されているが、システムの負荷が高いときに応答時間が長くなることはやむなしとされ、ユーザプロセスのＣＰＵ利用に関する保証はなかった。

【０００４】また、実時間性を必要とする組み込み機器の制御では、設計時に実行する全プロセスを把握することができるので、プロセスに固定した優先度をつけ、システムが優先度の高いプロセスを優先して実行する固定優先度スケジューラが用いられている。これに対し近年、マルチメディアシステムの発展により、ビデオや音声などを計算機のプロセススケジューラシステムで扱うようになってきている。ビデオや音声を滑らかに再生するためには、決まった時間間隔で一定の処理を確実に行わなければならない。そのようなマルチメディアシステムにとって、従来のスケジューリング処理は次のような問題を持っている。

【０００５】（１）従来のバッチあるいはタイムシェアリングのスケジューリングでは、ＣＰＵがいつどれだけ利用できるかということに関してなんら保証がない。このようなスケジューリングでは、マルチメディアアプリケーションプログラムが必要とする時にＣＰＵを利用できない。結果として、例えばビデオを再生したときにぎくしゃくした動きになってしまう。

【０００６】（２）マルチメディアアプリケーションプログラムは、組み込み機器と違い、どのようなプロセスあるいはプログラムが実行されるかを予測することは出来ない。このため、予めプロセスに優先順位を設定することができない。従って、従来の固定優先度のスケジューリングでは目的を果たせない。この問題は、周期的に動作するプロセスを、周期が短いプロセスほど優先するレート・モノトニック・スケジューリングであっても同じである。

【０００７】（３）マルチメディアアプリケーションプログラムは、組み込み機器と違い、プログラムの動作を信用できない。例えば、あるプログラムが故意に、または誤りから、ＣＰＵを永遠に使用してしまうと、固定優先度でスケジューリングしている場合、ＣＰＵを占有し

たプロセスより低い優先度のプロセスが全く動作しなくなってしまう。このような状況は、汎用のマルチメディアシステムでは許されない。

【０００８】以上のような理由により、現在、マルチメディアシステムで利用可能な新たなプロセススケジューリングが必要となり、その研究開発が進められている。

【０００９】

【発明が解決しようとする課題】マルチメディアシステムに適した新たなプロセススケジューラを導入するには、スケジューリングをオペレーティングシステム内で行う場合と、オペレーティングシステム上で動作するユーザプログラムで行う場合がある。しかし、従来のプロセススケジューラを導入するには、次のような問題があった。

【００１０】（１）オペレーティングシステム内に新しいプロセススケジューラを導入する場合の問題

（ａ）従来のオペレーティングシステムは、プロセススケジューラの追加や変更をオペレーティングシステム開発者以外には許していない。スケジューラを変更するには、オペレーティングシステムのソースコードや開発環境が必要である。商用オペレーティングシステムの場合、ソースコードを入手できなかったり、入手できても非常に高価なライセンス料が必要であるため、マルチメディアシステムの開発者が商用オペレーティングシステムのプロセススケジューラを変更することは事実上無理である。

【００１１】（ｂ）非商用オペレーティングシステムでは、ソースコードを無料で入手できる場合もあるが、商用オペレーティングシステム用に作られた多数のアプリケーションプログラムを動作させることができないし、サポート体制に問題があるなど、非商用オペレーティングシステムは計算機のプロセススケジューラの非専門家が利用できる状況にない。

【００１２】（ｃ）商用、非商用にかかわらず、オペレーティングシステムの内部を変更するには、オペレーティングシステムに関する一般的な知識と、そのオペレーティングシステムに固有の実装方法についての理解が必要であるが、そのような技術を持つものは限られている。

（２）ユーザプログラムでプロセススケジューラを導入する場合の問題

（ａ）ユーザプログラムやユーザプログラムが利用するライブラリで、自分のプロセス内に擬似的なプロセス（ユーザレベルスレッド）を実現し、ユーザレベルスレッドのスケジューリングを行う方法がある。この方法は、そのプロセス内のスケジューリングしか行えないため、システムに他のプロセス群が存在し、それらがオペレーティングシステムのスケジューリングのもとでＣＰＵを利用する場合には、マルチメディアシステムが必要としているＣＰＵ利用の保証は出来ない。

【００１３】（ｂ）ユーザプログラムがスケジューラとなり、このユーザレベルスケジューラが、オペレーティングシステムでサポートする固定優先度スケジューラで使用している優先度をユーザプロセスに与えるスケジューリング方法がある。この方法は、オペレーティングシステムが提供しているスケジューリングを利用するため、マルチメディアシステムが必要としているユーザプロセスでのＣＰＵ利用の保証はできない。

（ｃ）更に（ａ）（ｂ）では、一般にユーザプログラムの変更が必要になる。

【００１４】以上のように、従来のプロセススケジューラでは、マルチメディアシステムで必要としている新しいプロセススケジューラを導入することが非常に困難である。本発明では、オペレーティングシステムがサポートするスケジューラを利用してユーザプログラムでスケジューリングを行いながらも、マルチメディアシステムが必要としているユーザプロセスでのＣＰＵ利用が保証できる計算機のプロセススケジューラを提供することを目的とする。

【００１５】

【課題を解決するための手段】図１は本発明の原理説明図である。まず本発明は、オペレーティングシステム（ＯＳ）１４によりサポートされ、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定めると共に、優先度はユーザプロセスからの指定で変更可能であり、実行可能なプロセスのうち優先度の高いプロセスからＣＰＵを割当てて動作させる計算機のプロセススケジューラを前提とする。

【００１６】このような固定優先度のプロセススケジューラ、即ちリアルタイムクラス２８のプロセススケジューラ空間に、本発明は、図１（Ａ）のように、ユーザレベルプロセススケジューラ１０を設けたことを特徴とする。ユーザレベルプロセススケジューラ１０は、第１図（Ｂ）のように、リアルタイムクラス２８の第１優先度１５９を持ち、第１優先度１５９より低い優先度を持つ他のユーザプロセス群１１をスケジューリングして動作させる。即ち、ＣＰＵ割当てを決めてユーザプロセスの動作をオペレーティングシステム１４に依頼する。

【００１７】本発明の他の形態としては、第１優先度１５９をもつ複数のユーザプロセス１２毎にユーザレベルプロセススケジューラ１０をリンクさせ、自己のユーザプロセスをスケジューリングして動作させる分散型としてもよい。また本発明の他の形態としては、第１優先度１５９のユーザレベルプロセススケジューラ１０は、自己がスケジュールしているユーザプロセス群中に、第２優先度１５８のユーザレベルプロセススケジューラを設け、上位から下位に階層的に他のユーザプロセス群をスケジューリングして動作させる階層型としてもよい。この階層法は分散法に適用し、階層分散型としてもよい。

【００１８】さらに本発明の別の形態として、第一優先

度をもつユーザレベルスケジューラ１０を複数設け、各ユーザレベルプロセススケジューラ１０が個別に自己のユーザプロセス群１１をスケジューリングして実行するようにしてもよい。ユーザレベルプロセススケジューラ１０は、図１（Ａ）のように、ユーザプロセスからの要求に基づくクラス変更指示部１６からの指示、又はプロセス管理表１８の参照によるプロセス実行指示部２０からの指示に従い、指定されたユーザプロセスの実行、停止又は再開を、図１（Ｂ）のように、オペレーティングシステム１４に依頼するプロセス実行制御部２４を備える。

【００１９】またユーザレベルプロセススケジューラ１０は、図１（Ａ）のように、ユーザプロセスからの要求に基づくクラス変更指示部１６からの指示、又はプロセス管理表１８の参照によるプロセス実行指示部２０からの指示に従い、指定されたユーザプロセスＡの優先度を第１優先度に変更して実行させ、他のユーザプロセスＢを下位の優先度に変更して停止させるように、ユーザプロセスの実行、停止、又は再開を図１（Ｃ）のようにオペレーティングシステム１４に依頼するプロセス優先度制御部２６を添える。

【００２０】更に、オペレーティングシステム１４は、他界優先度の範囲（１００～１５９）にある固定優先度のリアルタイムクラス２８に対し、低い優先度の範囲（０～５０）にあってプロセス状態に応じて動的に変化させる他のスケジューリングクラス、例えばタイムシェアリングクラス３０を有する場合、ユーザレベルプロセススケジューラ１０は、ユーザプロセスからの要求に基づくクラス変更指示部１６からの指示、又はプロセス管理表１８の参照によるプロセス実行指示部２０からの指示に従い、指定されたクラスの変更をオペレーティングシステム１４に依頼して、図１（Ｄ）のように、ユーザプロセスの実行、停止又は再開させるクラス変更制御部２４を備える。

【００２１】ここで、クラス変更指示部１６は、ユーザプロセスＣのＣＰＵ使用時間の指定を伴なう加入要求に対し、プロセス管理表１８を参照して現在加入状態にあるユーザプロセスＡ，Ｂの要求ＣＰＵ時間の合計時間を求め、この合計時間が規定時間以下であればユーザプロセスＣの加入を許可し、規定値を超えていれば加入を拒否する。

【００２２】加入処理は、クラス変更指示部１６がユーザプロセスＣの加入を許可する場合、プロセス実行制御部２４に加入要求プロセスＣの実行停止状態への変更を指示し、且つクラス変更制御部２２にリアルタイムクラスへの変更を指示し、更にプロセス優先度制御部２６に第２優先度への変更を指示し、更にユーザプロセスＣをプロセス管理表１８に登録した後に、加入許可を通知する。

【００２３】また加入処理の別の形態として、クラス変更指示部１６は、ユーザプロセスＣの加入を許可する場合、クラス変更制御部２２にタイムシェアリングクラス３０からリアルタイムクラス２８への変更を指示し、且つプロセス優先度制御部２６にリアルタイムクラスの例えば最下位優先度１００への変更を指示し、更にユーザプロセスＣをプロセス管理表１８に登録した後に、加入許可を通知する。

【００２４】更に、加入処理の別の形態としてクラス変更指示部１６は、ユーザプロセスＣの加入を許可する場合、クラス変更制御部２２にタイムシェアリングクラス３０への変更を指示し、更にユーザプロセスＣをプロセス管理表１８に登録した後に、加入許可を通知する。ユーザプロセスの脱会要求があると、クラス変更指示部１６は、プロセス管理表１８からユーザプロセスを削除すると共に、クラス変更制御部２２に脱会要求を行ったユーザプロセスのタイムシェアリングクラス３０への変更を指示する。

【００２５】クラス変更指示部１６は、ユーザプロセスからの通知により新規なユーザプロセスの加入と脱会を処理できるし、またオペレーティングシステム１４からの通知により新規なユーザプロセスの加入と脱会を処理することもできる。更に、ユーザレベルプロセススケジューラ１０のプロセス実行制御部２４、クラス変更制御部２２、プロセス優先度制御部２６に対する変更要求の情報をライブラリに格納し、アプリケーションからの要求指示でライブラリからユーザレベルプロセススケジューラ１０に、対応する変更要求のメッセージを通知してオペレーティングシステムに変更を依頼することもできる。

【００２６】またプロセス実行制御部２４、クラス変更制御部２２、プロセス優先度制御部２６に対する変更要求のスケジューリング仕様をユーザプロセスの１つで生成し、ユーザプロセスのスケジューリング仕様に基づきユーザレベルプロセススケジューラ１０に変更要求を通知して、オペレーティングシステム１４に変更を依頼する。

【００２７】ここで、ユーザレベルプロセススケジューラ１０のスケジューリング対象となるユーザプロセス１２は、複数のスレッドから構成され、ユーザプロセスの一部であるスレッドスケジューラにより複数のスレッドをスケジューリングする。ユーザプロセスのユーザレベルプロセススケジューラ１０は、プロセス管理表１８に登録された１又は複数のユーザプロセスにＣＰＵ時間を割り当てる際、ＣＰＵ割当対象となるユーザプロセスが１つしか存在しないように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステム１０に依頼する。

【００２８】またユーザレベルプロセススケジューラ１０は、１又は複数のユーザプロセスにＣＰＵ時間を割り当てる際、同じＣＰＵ割当時間帯に複数のユーザプロセ

スが時分割で存在するように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステム１４に依頼する。ユーザレベルプロセススケジューラ１０は、一定期間内に複数のユーザプロセスと自分自身を含めてＣＰＵ時間を順番に割り当てることにより、複数のユーザプロセスの動作を一定期間内で保証する。またユーザレベルプロセススケジューラ１０は、一定の周期を設定し、各周期内で複数のユーザプロセスと自分自身を含めてＣＰＵ時間を順番に割り当てることを繰り返すことにより、複数のユーザプロセスで動作を保証する。

【００２９】複数のユーザプロセスが持つ動作周期が異なる場合、ユーザレベルプロセススケジューラ１０は、異なる動作周期の組合わせにより１つの周期を決定し、この周期内で複数のユーザプロセスに使用可能なＣＰＵ時間を順番に割り当て、複数のユーザプロセスの動作を保証する。ユーザレベルプロセススケジューラ１０は、一定時間又は一定周期において複数のユーザプロセスに割り当てたＣＰＵ時間の占める割合が規定値以内の時、他のユーザプロセスに対するＣＰＵ時間の割当てを許容し、規定値を越えた時、他のユーザプロセスに対するＣＰＵ時間の割当てを拒否するアドミッションコントロールを行う。

【００３０】ユーザレベルプロセススケジューラ１０は、一定期間又は一定周期でユーザプロセスにＣＰＵ時間を割り当てる場合、連続したＣＰＵ時間を割り当ててもよいし、複数のユーザプロセスのＣＰＵ時間の各々を分割して割り当ててもよい。プロセス管理表１８に登録されたユーザプロセスとして、高い優先度の範囲（１００～１５９）にある固定優先度のリアルタイムクラス２８に属するユーザプロセスと、低い優先度の範囲（０～５０）にあって動的に優先度を変化させるタイムシェアリングクラスに属するユーザプロセス３０が存在する場合、ユーザレベルプロセススケジューラ１０は、リアルタイムクラス２８のユーザプロセスに対するＣＰＵ時間の割当てを制限して、タイムシェアリングクラス３０のユーザプロセスにＣＰＵ時間を割り当てる。

【００３１】ユーザレベルプロセススケジューラ１０は、ユーザプロセスの実行停止を検出して他のユーザプロセスを動作可能とする。ユーザレベルプロセススケジューラ１０は、ユーザプロセスにＣＰＵ時間を割り当てる際に、ユーザプロセスに加え同じ優先度で実行可能なブロック検出プロセスを設け、ユーザプロセスがＩ／Ｏ待ち等でブロックした際に、ブロック検出プロセスにＣＰＵ時間を割り当ててブロック発生を検出して通知させ、この通知によりユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とする。

【００３２】またユーザレベルプロセススケジューラ１０は、ユーザプロセスにＣＰＵ時間を割り当てる際に、更に、ユーザプロセスより低い優先度で実行可能なブロック検出プロセスを設け、ユーザプロセスがＩ／Ｏ待ち等でブロックした際にブロック検出プロセスにＣＰＵ時間を割り当ててブロック発生を検出して通知させ、この通知によりユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とする。

【００３３】ユーザレベルプロセススケジューラ１０は、ブロック中のユーザプロセスが実行可能な状態になったことを検出して動作させる。ユーザレベルプロセススケジューラ１０は、オペレーティングシステム１４によるユーザプロセスのブロックまたはブロック回復による実行再開の通知を検出して他のユーザプロセスの実行又は停止を行うこともできる。

【００３４】このような本発明のユーザレベルプロセススケジューラによれば、次の利点が得られる。

（１）スケジューリング対象のプロセスに、常に一定割合のＣＰＵを割り当て、ＣＰＵ利用を保証するようなスケジューラを実現できる。

（２）従来のプログラムを変更せずにユーザプロセスをスケジューリング対象とすることができる。新しいスケジューラのためにプログラムを書き直したり、リンクし直したりする必要がない。

【００３５】（３）新たなスケジューラを実現するのにオペレーティングシステムのソースコードを必要としない。

（４）商用のオペレーティングシステム上で利用できる。

（５）新たなスケジューリングを実現するために、オペレーティングシステム内部の知識を必要としない。

【００３６】（６）各種のスケジューラを試す時に、システムを再起動する必要がない。

（７）振る舞いが信用できないプログラムをスケジューリングの対象としても、他のプログラムの実行を進めることのできるスケジュールを実現できる。この結果、マルチメディア処理で必要とされる一定割合のＣＰＵ利用を保証するスケジューラなど、ユーザのニーズに合わせてさまざまなスケジューラを商用オペレーティングシステムの上に実装できる。

【００３７】また、振る舞いの信用できないプロセスをスケジューリング対象とすることができるなど、オペレーティングシステム内で実現するのと同等のスケジューラをユーザレベルのスケジューラとして実装することができる。

【００３８】

【発明の実施の形態】

＜目　次＞

１．動作環境

本発明のユーザレベルプロセススケジューラは、オペレーティングシステムが固定優先度のスケジューリングを提供していることを前提に実現することができる。固定優先度スケジューリングは、組込用のリアルタイムオペレーティングシステムでは以前から提供されており、また最近の汎用オペレーティングシステムとして知られているＵＮＩＸ　Ｓｙｓｔｅｍ　Ｖ、Ｓｕｎ　ＯＳ５、Ｗｉｎｄｏｗｓ／ＮＴでも提供されるようになってきている。

【００３９】オペレーティングシステムの標準を規定するＰＯＳＩＸ（ユニックス標準の統合組織）では、固定優先度スケジューリングを行うスケジューラを規定している。サンマイクロシステム社のオペレーティングシステムＳｏｌａｒｉｓ２．４は、ＰＯＳＩＸに準拠しており、リアルタイムクラスと呼ばれる固定優先度スケジューラをもっている。このリアルタイムクラスでは、優先度として１００～１５９の値を使用することができる。優先度は、値が大きい方が優先して実行されることを意味する。以下の実施例にあっては、特に断わらない限り、オペレーティングシステムＳｏｌａｒｉｓ２．４上での動作環境を例にとるものとする。

【００４０】図２は本発明のユーザレベルプロセススケジューラの動作環境の実施形態である。図２において、オペレーティングシステムで提供される固定優先度スケジューリングのリアルタイムクラスの空間において、本発明のユーザレベルプロセススケジューラ１０はリアルタイムの最高優先度である優先度１５９で動作するように優先度が設定され、スケジューリング対象となるユーザプロセス群１１のユーザプロセス１２－１，１２－２を第２優先度となる優先度１５８で動作させるように設定している。

【００４１】図３は本発明によるユーザレベルプロセススケジューラの動作環境の他の実施形態であり、この実施形態にあっては、リアルタイムクラスの優先度１５９で動作するユーザレベルプロセススケジューラは、ユーザレベルプロセススケジューラ１０－１，１０－２の２つに分割されている。ユーザレベルプロセススケジューラ１０－１，１０－２のそれぞれは、スケジューリング対象であるユーザプロセス１２－１，１２－２にライブラリとしてそれぞれリンクされている。このためユーザレベルプロセススケジューラ１０－１，１０－２は、それぞれリンクされているユーザプロセス１２－１，１２－２の動作を制御する。

【００４２】図４は本発明によるユーザレベルプロセススケジューラの他の実施形態であり、この実施形態にあっては、ユーザレベルプロセススケジューラが階層的にスケジューリングを行うことを特徴とする。まずユーザレベルプロセススケジューラ１０－１１は、リアルタイムクラスの第１優先度である優先度１５９で動作するように優先度が設定される。

【００４３】優先度１５９を設定したユーザレベルプロセススケジューラ１０－１１のスケジューリング対象として、第２優先度となる優先度１５８を設定した第２のユーザレベルプロセススケジューラ１０－１２及びユーザプロセス１２－１を設けて、それぞれ動作させている。更に、第２のユーザレベルプロセススケジューラ１０－１２は、スケジューリング対象として優先度１５７を設定したユーザプロセス１２－２，１２－３をもち、それぞれ優先度１５７で動作させる。

【００４４】図５はユーザレベルプロセススケジューラによる階層的なスケジューリングの他の実施形態であり、この実施形態にあっては、図３の実施形態について階層的なスケジュール構造としたことを特徴とする。即ち、リアルタイムクラスの優先度１５９で動作するユーザプロセス１２－１，１２－２のそれぞれにライブラリとしてリンクされたユーザレベルプロセススケジューラ１０－１，１０－２は、優先度１５８を設定して動作するユーザプロセス１２－３，１２－４にライブラリとしてリンクしたユーザレベルプロセススケジューラ１０－３，１０－４を設けている。

【００４５】即ち、ユーザレベルプロセススケジューラ１０－１，１０－２は、第１階層となる優先度１５９でユーザプロセス１２－１，１２－２をスケジューリングし、ユーザプロセス１２－１，１２－２のスケジュール方針に従って、第２階層となる優先度１５８のユーザレベルプロセススケジューラ１０－３，１０－４がユーザプロセス１２－３，１２－４をスケジューリングすることになる。

【００４６】図６は本発明によるユーザレベルプロセススケジューラの他の実施形態であり、図２の実施形態をリアルタイムクラスについて複数系統設けたことを特徴とする。即ち、リアルタイムクラスの優先度１５９で動作する２つのユーザレベルプロセススケジューラ１０－１，１０－２を設けると共に、それぞれのスケジューリング対象となるユーザプロセス群１１－１，１１－２として、優先度１５８で動作するユーザプロセス１２－１，１２－２及びユーザプロセス１２－３，１２－４を設けている。このような実施形態にあっては、ユーザレベルプロセススケジューラ１０－１がユーザプロセス１２－１，１２－２をスケジューリングし、またユーザレベルプロセススケジューラ１０－２がユーザプロセス１２－３，１２－４をスケジューリングすることになる。もちろん図６はオペレーティングシステム内に２つのユーザレベルプロセススケジューラ１０－１，１０－２が存在する場合であるが、必要に応じて任意の数のユーザレベルプロセススケジューラを存在させることができる。

２．ユーザレベルプロセススケジューラの機能と動作

次に図２の実施形態を対象に、本発明により提供される

ユーザレベルプロセススケジューラの機能と動作を説明する。図７は、リアルタイムクラスの第１優先度となる優先度１５９で動作するユーザレベルプロセススケジューラ１０の詳細である。ユーザレベルプロセススケジューラ１０は、クラス変更指示部１６、プロセス管理表１８、プロセス実行指示部２０、クラス変更制御部２２、プロセス実行制御部２４及びプロセス優先度制御部２６で構成される。

【００４７】クラス変更指示部１６は、ユーザレベルプロセススケジューラ１０が動作しているスケジュール空間であるリアルタイムクラス２８以外のスケジューリングクラス、例えばタイムシェアリングクラス３０で動作しているユーザプロセス１２－３からの加入要求を受け、プロセス管理表１８を参照して、予め定めた条件に従って許可または拒否を決定する。

【００４８】許可の場合にはクラス変更制御部２２にタイムシェアリングクラス３０からリアルタイムクラス２８へのクラス変更を指示し、ユーザプロセス１２－３に許可を通知する。加入要求に対する許可の条件としては、例えばプロセス管理表１８で管理しているユーザプロセス群の合計ＣＰＵ時間である。ユーザプロセス１２－３は加入要求と同時に自己のＣＰＵ時間を指定し、この指定時間を現在プロセス管理表１８で管理しているユーザプロセスのＣＰＵ時間に加えて、規定時間以内であれば加入を許可し、超えていれば加入を拒否するようになる。

【００４９】ここでプロセス管理表１８は、図８に示すように、プロセス識別子と例えば動作可能状態としたユーザプロセスを一定周期内で動作させる場合、１周期内のＣＰＵ使用時間を登録している。１周期内のＣＰＵ使用時間は、例えば１００％で１００ｍｓとしている。この例では、図７のユーザプロセス１２－１，１２－２，１２－３のプロセス識別子をＡ，Ｂ，Ｃとすると、ユーザプロセスＣ１２－３が加入要求を行ったときに、既にプロセス管理表１８には動作可能状態にあるユーザプロセス１２－１，１２－２のプロセス識別子Ａ，Ｂとその１周期内のＣＰＵ使用時間５０ｍｓ，２０ｍｓが登録されている。

【００５０】この状態でプロセス識別子Ｃをもつユーザプロセス１２－３の加入要求があると、加入要求に伴ってＣＰＵ使用時間１０ｍｓが指定されていることから、合計ＣＰＵ使用時間として８０ｍｓが求められ、これは全ＣＰＵ使用時間の８０％であることから加入要求を許可し、プロセス管理表１８にプロセス識別子ＣとそのＣＰＵ使用時間１０ｍｓを図示のように登録することになる。

【００５１】再び図７を参照するに、プロセス実行指示部２０はプロセス管理表１８を参照し、次にどのプロセスをどれだけの時間実行するかを決定し、クラス変更制御部２２に変更のための指示を行う。即ちプロセス実行指示部２０は、クラス変更制御部２２に対しては、タイムシェアリングクラス３０とリアルタイムクラス２８との間のクラス変更の指示を行う。

【００５２】またプロセス実行指示部２０は、プロセス実行制御部２４に対しては、実行、停止または再開の指示を行う。更にプロセス実行指示部２０は、プロセス優先度制御部２６に対しては、リアルタイムクラス２８における優先度１５９～１００内における変更を指示する。クラス変更制御部２２は、クラス変更指示部１６またはプロセス実行指示部２０からのクラス変更の指示に従い、指定されたユーザプロセスのクラス変更をオペレーティングシステム１４に依頼する。このクラス変更の依頼を受けて、オペレーティングシステム１４は、固定優先度スケジュールクラスであるリアルタイムクラス２８と、それ以外のスケジュールクラスであるタイムシェアリングクラス３０との間のクラス変更を実行する。

【００５３】プロセス実行制御部２４は、クラス変更指示部１６またはプロセス実行指示部２０の指示に従い、指定されたユーザプロセスの実行、停止または再開をオペレーティングシステム１４に依頼する。更にプロセス優先度制御部２６は、クラス変更指示部１６またはプロセス実行指示部２０の指示に従い、固定優先度であるリアルタイムクラス２８における優先度の変更をオペレーティングシステム１４に依頼する。

【００５４】ユーザレベルプロセススケジューラ１０に設けたクラス変更制御部２２、プロセス実行制御部２４及びプロセス優先度制御部２６は、そのときのクラス変更指示部１６またはプロセス実行指示部２０からの指示により個別に動作して、ユーザプロセスのスケジューリングの変更を行う。そこで、これら３つの制御部２２，２４，２６のそれぞれについてユーザプロセスの変更制御を説明する。

【００５５】図９は図７のプロセス実行制御部２４によるユーザプロセスの状態変更の実施形態であり、オペレーティングシステム１４が提供するシグナル機能を用いてユーザプロセスの状態変更を行うことで、ユーザレベルプロセススケジューラがユーザプロセスの実行制御を行うことを特徴とする。図９において、優先度１５９のユーザレベルプロセススケジューラ１０は、スケジューリング対象としてユーザ部１１にプロセス識別子Ａ，Ｂをもったユーザプロセス１２－１，１２－２をもっている。このユーザプロセス１２－１，１２－２の実行可能状態への変更は、オペレーティングシステム１４が提供するシグナル機能であるＳＩＧＣＯＮＴを引数としてｋｉｌｌシステムコールを発行することで実現できる。

【００５６】またユーザプロセス１２－１，１２－２の実行停止状態への変更は、ＳＩＧＳＴＯＰを引数としてｋｉｌｌシステムコールを発行することで実現できる。例えば図９のように、ユーザレベルプロセススケジューラ１０がプロセス識別子Ａをもつユーザプロセス１２－

１を実行可能状態とし、プロセス識別子Ｂをもつユーザプロセス１２－２を実行停止状態とすることで、ユーザプロセス１２－１だけを動作させることができる。

【００５７】図１０のフローチャートは、図７のプロセス実行指示部２０からの実行指示に基づくプロセス実行制御部２４の処理動作である。プロセス実行制御部２０は、例えばオペレーティングシステム１４からのＳＩＧＣＯＮＴを引数としたｋｉｌｌシステムコールを受けると、まずステップＳ１で、プロセス管理表１８のインデックスを先頭のエントリとする。

【００５８】ステップＳ２で、インデックスの示すエントリは現在使用中（登録中）にあるか否かチェックする。使用中にあれば、ステップＳ３で、インデックスの示すプロセス識別子のユーザプロセスの実行状態への変更をプロセス実行制御部２４に指示する。例えば図８のプロセス管理表１８の場合には、最初のインデックスで指定されるユーザプロセスＡの実行をプロセス実行制御部２４に指示する。

【００５９】このためプロセス実行制御部２４は、ステップＳ２４でユーザプロセスＡのＣＰＵ使用時間である５０ｍｓ間スリープすることで、ユーザプロセスＡを５０ｍｓ間と動作させる。続いてステップＳ５で、インデックスの示すユーザプロセスＡの実行停止状態への変更をプロセス実行制御部２４に依頼し、これを受けてオペレーティングシステム１４はユーザプロセスＡの実行を停止する。ステップＳ３～Ｓ４で最初のインデックスのユーザプロセスＡの動作が済むと、ステップＳ６で、インデックスが最後のエントリを指しているか否かチェックし、指していなければ、ステップＳ７でインデックスを１つ進め、ステップＳ２に戻り、次の２番目インデックスの示すエントリのユーザプロセスＢの処理を同様にして行う。

【００６０】ステップＳ６でインデックスの最後のエントリが判別されると、再びステップＳ１に戻り、プロセス管理表１８の先頭のインデックスからの処理を繰り返す。これにより１周期内でプロセス管理表１８に登録されて動作可能状態にある複数のユーザプロセスを、設定されたＣＰＵ時間で動作させるスケジューリングを行うことができる。

【００６１】図１１は、図７のプロセス優先度制御部２６によるユーザプロセスの変更の実施形態である。この実施形態にあっては、オペレーティングシステム１４が提供するｐｒｉｏｃｎｔｌシステムコールを使用し、ユーザプロセスの優先度を変更することで、ユーザレベルプロセススケジューラがユーザプロセスの実行制御を行うことを特徴とする。

【００６２】例えば図１１のように、ユーザレベルプロセススケジューラ１０が優先度１５７にあるプロセス識別子Ａのユーザプロセス１２－１を優先度１５８に変更することで、プロセス識別子Ａのユーザプロセス１２－１だけを動作させることができる。図１２は、図７のプロセス実行指示部２０からの優先度変更指示に基づくプロセス優先度制御部２６の処理動作を示す。

【００６３】まずステップＳ１で、優先度変更指示に対しプロセス管理表１８のインデックスを先頭にセットし、ステップＳ２で、インデックスの示すエントリが使用中か否かチェックする。使用中にあれば、ステップＳ３で、インデックスで示すユーザプロセスの優先度１５８への変更をプロセス優先度制御部２６に指示する。ステップＳ４で、インデックスが示すユーザプロセスのＣＰＵ時間をスリープすることで、ユーザプロセスＣを指定時間動作させる。更にステップＳ５で、インデックスの示すユーザプロセスの優先度１５７への変更をプロセス優先度制御部２６に指示する。なおステップＳ５にあっては、停止動作を確実にするため、リアルタイムクラスの最下位の優先度である優先度１００への変更をプロセス優先度制御部２６に指示するようにしてもよい。

【００６４】続いてステップＳ６で、インデックスが最後のエントリを指しているか否かチェックし、指していなければ、ステップＳ７でインデックスを１つ進め、再びステップＳ２に戻り、プロセス管理表の次のインデックスのエントリに登録されているユーザプロセスについて、ステップＳ３～Ｓ５の優先度の変更による実行と停止を繰り返す。

【００６５】図１３は、図７のプロセス実行指示部２０からのクラス変更指示に基づくクラス変更制御部２２のユーザプロセスの変更処理の実施形態である。本発明が実現されるオペレーティングシステムＳｏｌａｒｉｓ２．４は、固定優先度のリアルタイムクラス２８に加えてタイムシェアリングスケジューリングを行うタイムシェアリングクラス３０をもっている。タイムシェアリングクラス３０にあっては、オペレーティングシステム１４が次のようにユーザプロセスの優先度を動的に変更する。ここでタイムシェアリングクラス３０は、優先度５９～０をもつ。

【００６６】（１）ユーザプロセスが一定時間ＣＰＵを使用した場合、その優先度を一定値下げる。

（２）ユーザプロセスが実行可能であるにも関わらず一定時間ＣＰＵ割当てが行われなかった場合、その優先度を一定値上げる。

（３）ユーザプロセスが自発的にＣＰＵ割当てを手放した場合、即ちスリープした場合、ウェイクアップ後にその優先度を一定値上げる。

【００６７】このようなタイムシェアリングクラス３０と本発明のユーザレベルプロセススケジューラが動作しているリアルタイムクラス２８のクラス変更として、オペレーティングシステム１４が提供するｐｒｉｏｃｎｔ１システムコールを使用してクラス変更を行うことで、ユーザレベルプロセススケジューラがユーザプロセスの実行制御を行うことができる。

【００６８】例えば図１３のように、ユーザレベルプロセススケジューラ１０がタイムシェアリングクラスで動作していたプロセス識別子Ａのユーザプロセス１２－１をリアルタイムクラス２８の優先度１５８に変更することで、プロセス識別子Ａのユーザプロセス１２－１´のみを動作させることができる。図１４のフローチャートは、図７のプロセス実行指示部２０よりクラス変更指示をクラス変更制御部２２に対し行ったときのユーザプロセスの変更処理である。

【００６９】オペレーティングシステムが提供するｐｒｉｏｃｎｔｌシステムコールを使用してプロセス実行指示部２０に対しクラス変更指示が行われると、まずステップＳ１でプロセス管理表１８のインデックスを先頭にセットし、ステップＳ２で、インデックスの示すエントリが使用中か否かチェックする。使用中にあれば、ステップＳ３で、インデックスの示すユーザプロセスのリアルタイムクラス２８への変更をクラス変更指示部２２に指示する。

【００７０】これによりオペレーティングシステム１４は、対象となったユーザプロセスをリアルタイムクラスの優先度１５８に変更し、ステップＳ４で、インデックスが示すユーザプロセスのＣＰＵ時間をスリープすることで、ユーザプロセスを指定時間動作させる。ステップＳ５で、インデックスの示すユーザプロセスのタイムシェアリングクラス２８への変更をクラス変更制御部２２に指示し、停止状態とする。

【００７１】続いてステップＳ６で、インデックスが最後のエントリを指しているか否かチェックし、指していなければ、ステップＳ７でインデックスを１つ進め、ステップＳ２に戻って次のユーザプロセスについて同様な処理を繰り返す。

３．ユーザプロセスの加入脱会

図７のリアルタイムクラス２８のスケジュール空間にあるユーザレベルプロセススケジューラ１０のスケジュール対象となるユーザプロセスへの加入は、タイムシェアリングクラス３０に存在するユーザプロセスからの加入要求を受けて処理するようになる。またリアルタイムクラス２８でスケジューリングされているユーザプロセスの脱会は、ユーザプロセスからの脱会要求を受けて処理するようになる。

【００７２】このユーザプロセスの加入要求及び脱会要求の処理はクラス変更指示部１６に対し行われ、クラス変更指示部１６は、クラス変更制御部２２、プロセス実行制御部２４またはプロセス優先度制御部２６のいずれかに対する指示で３つの異なった加入脱会処理を行うことができる。図１５のフローチャートは、プロセス実行制御部２４に対する実行、停止または再開の指示を利用したユーザプロセスの加入脱会処理である。まずステップＳ１で、タイムシェアリングクラス３０にあるユーザプロセスからの加入または脱会要求を待っている。ステップＳ２で、要求があると加入要求か否かチェックする。

【００７３】いまタイムシェアリングクラス３０にあるプロセス識別子Ｃのユーザプロセス１２－３から加入要求が行われたとする。この実施形態にあっては、ユーザプロセス１２－３は加入要求に際し自分自身の１周期内のＣＰＵ使用時間を例えば１０ｍｓとして指定して加入を要求している。加入要求を受けたことでステップＳ３に進み、クラス変更指示部１６はプロセス管理表１８に登録されているユーザプロセスのＣＰＵ使用時間の合計と加入要求プロセスの要求ＣＰＵ時間の和が、予め定めた所定時間１００ｍｓより小さいか否かチェックする。

【００７４】ここで図８のプロセス管理表１８のように、プロセス識別子Ａ，Ｂのユーザプロセス１２－１，１２－２が登録されており、それぞれの１周期内のＣＰＵ時間が５０ｍｓ，２０ｍｓであったとすると、加入要求を行ったユーザプロセス１２－３のＣＰＵ使用時間１０ｍｓを加えた合計時間は８０ｍｓとなり、１００ｍｓより小さいことから加入を許可するため、ステップＳ４に進む。

【００７５】ステップＳ４では、加入要求を行ったユーザプロセスの実行停止状態への変更をプロセス実行制御部２４に指示し、オペレーティングシステム１４への依頼で、加入要求を行ったユーザプロセス１２－３を停止状態とする。続いて、加入要求を行ったユーザプロセス１２－３のリアルタイムクラス２８への変更をステップＳ５でクラス変更制御部２２に指示する。

【００７６】更にステップＳ６で、加入要求を行ったユーザプロセス１２－３への優先度１５８への変更をプロセス優先度制御部２６に指示する。これにより加入要求を許可されたユーザプロセス１２－３は、ユーザレベルプロセススケジューラ１０が動作しているリアルタイムクラス２８の優先度１５８に設定される。続いてステップＳ７で、加入要求を行ったユーザプロセス１２－３を図８のプロセス識別子ＣとそのＣＰＵ時間１０ｍｓに示すようにプロセス管理表１８に登録する。

【００７７】最終的にステップＳ５で、加入要求を行ったユーザプロセス１２－３に対し加入許可を通知する。一方、ステップＳ３で加入要求を行ったユーザプロセスを含めた合計ＣＰＵ時間が１００ｍｓを超えていた場合には、ステップＳ９に進み、加入要求を行ったユーザプロセス１２－３に対し加入拒否を通知する。またユーザプロセスからの脱会要求があった場合には、ステップＳ２からＳ１０に進み、脱会要求を行ったユーザプロセスをプロセス管理表１８から削除し、ステップＳ１１で、脱会要求をしたユーザプロセスのタイムシェアリングクラス３０への変更をクラス変更制御部２２に指示する。

【００７８】図１６のフローチャートは、ユーザプロセスからの加入要求に対しプロセス優先度制御部２６による優先度の変更制御を利用して加入させる場合の処理動

作である。まずステップＳ１でユーザプロセスからの加入または脱会要求を待ち、要求があるとステップＳ２で加入要求か否か判別し、加入要求であれば、ステップＳ３で、プロセス管理表１８の登録済みユーザプロセスのＣＰＵ使用時間と加入要求プロセスの要求ＣＰＵ時間との和が１００ｍｓより小さいか否かチェックし、小さければ加入許可のため、ステップＳ４に進む。

【００７９】ステップＳ４では、加入要求プロセスのリアルタイムクラス２８への変更をクラス変更制御部２２に指示し、ステップＳ５で、加入要求プロセスの優先度１００への変更をプロセス優先度制御部２６に指示し、ステップＳ６で、加入要求をしたユーザプロセスをプロセス管理表１８に登録した後、ステップＳ７で加入許可を通知する。ステップＳ３で１００ｍｓ以上であった場合は、ステップＳ７で加入拒否を通知する。

【００８０】またステップＳ２で脱会要求であった場合には、ステップＳ９でプロセス管理表１８からの削除と、ステップＳ１０のタイムシェアリングクラス３０へのクラス変更の指示を行う。図１７のフローチャートは、ユーザプロセスからの加入または脱会要求に対し図７のクラス変更制御部２２の制御機能を利用して行う処理動作である。ステップＳ１でユーザプロセスからの加入または脱会要求を待ち、要求があると、ステップＳ２で加入要求か否か判別し、加入要求であればステップＳ３に進み、プロセス管理表１８のＣＰＵ使用時間と加入要求プロセスの要求ＣＰＵ時間の合計が１００ｍｓより小さいか否かチェックする。

【００８１】合計時間が１００ｍｓより小さければ加入許可のため、ステップＳ４に進み、加入要求を行ったユーザプロセスのタイムシェアリングクラス３０への変更をクラス変更制御部２２に指示する。続いてステップＳ５で、加入要求を行ったユーザプロセスをプロセス管理表１８に登録し、ステップＳ６で加入許可を通知する。この場合には、加入要求を行ったユーザプロセスは最初タイムシェアリングクラス３０に置かれて、実行の際にユーザレベルプロセススケジューラ１０によるリアルタイムクラス２８にクラス変更されることになる。

【００８２】一方、ステップＳ３で１００ｍｓ以上であった場合には、ステップＳ７で加入拒否を通知する。また脱会要求であった場合には、ステップＳ８でプロセス管理表１８から削除した後、ステップＳ９で、タイムシェアリングクラス３０へのクラス変更を指示する。図１８は、ライブラリによるユーザレベルプロセススケジューラに対するスケジューリング及び優先度変更要求の通知の実施形態である。

【００８３】図１８において、アプリケーションプログラム３２にリンクされるライブラリ３４にユーザレベルプロセススケジューラ１０のスケジューリング方式、即ちリアルタイムクラスかタイムシェアリングクラスかの指定情報、及びまたは優先度を変更する関数を用意する。このライブラリ３４の内容が呼び出された場合、メッセージによりユーザレベルプロセススケジューラ１０に変更を通知する。メッセージによる変更通知を受けたユーザレベルプロセススケジューラ１０は、変更が可能かどうかを判断し、可能ならばオペレーティングシステムから提供されるｐｒｉｏｃｎｔｌシステムコールにより変更する。変更が不可能ならばその旨をライブラリ３４に通知し、更にライブラリ３４がアプリケーションプログラム３２に通知する。

【００８４】図１９は、ユーザレベルプロセススケジューラに対するライブラリによるスケジューリング方式及びまたは優先度変更要求の通知の他の実施形態である。この図１９の実施形態にあっては、ユーザプロセス１２を生成する親プロセス３６が、リンクするライブラリ３８を通してユーザレベルプロセススケジューラ１０に、予め定めたスケジューリング仕様に従ったメッセージを伝えてスケジューリング方式及び優先度変更要求を行うようにしている。

【００８５】親プロセス３６で作成されるスケジューリング仕様４０は、例えば図２０に示すように、ユーザプロセス１２が動作し始めてからの相対的な開始時刻、終了時刻、及びその間のスケジューリング方式を記述している。このようなスケジューリング仕様４０に従ったライブラリ３８からのメッセージを受けたユーザレベルプロセススケジューラ１０は、スケジューリング仕様４０に従ったユーザプロセス１２のスケジューリングを実行することになる。

【００８６】図２１は、本発明のユーザレベルプロセススケジューラでスケジューリングされるユーザプロセスの内部構造を示している。本発明が対象とするユーザレベルプロセススケジューラのスケジューリング対象となるユーザプロセス１２は、図２１のようにマルチスレッドライブラリ４２をリンクしており、スレッド群４４の中に例えばスレッド４６－１，４６－２，４６－３の３つのスレッドがマルチスレッドライブラリ４２によりスケジューリングされている。

４．ユーザプロセスのＣＰＵ利用の保証

図７に示す制御機能をもつユーザレベルプロセススケジューラ１０は、スケジューリング対象となっているユーザプロセス即ちプロセス管理表１８に登録されているユーザプロセスを対象に、一定割合のＣＰＵ割当てを行ってＣＰＵ利用を保証するようなスケジューリングを実行できる。

【００８７】図２２は、ＣＰＵ割当て対象となるユーザプロセスを１つとした場合の動作形態の説明図である。図２２にあっては、ユーザレベルプロセススケジューラ１０が動作したいプロセス識別子Ａのユーザプロセス１２－１を実行可能状態に変更する指示を行う。この変更指示はプロセス実行指示部２０により行われ、プロセス実行制御部２４に対する実行／停止指示、クラス変更制

御部２２に対するクラス変更指示、またはプロセス優先度制御部２６に対する優先度変更指示のいずれかとなる。

【００８８】プロセス実行指示部２０によるプロセス実行制御部２４に対する指示については、ｋｉｌｌシステムコールによりユーザプロセス１２－１だけを実行状態とし、その他のユーザプロセスを実行停止状態とする。またプロセス実行指示部２０からのプロセス優先度制御部２６に対する優先度変更指示については、ｐｒｉｏｃｎｔｌシステムコールによりユーザプロセス１２－１だけを優先度１５８に設定し、その他のユーザプロセスを優先度１５７に設定する。

【００８９】更に、プロセス実行指示部２０によるクラス変更制御部２２に対するクラス変更指示については、ｐｒｉｏｃｎｔｌシステムコールによりユーザプロセス１２－１だけをリアルタイムクラス２８に変更し、その他のユーザプロセスをタイムシェアリングクラス３０に変更する。以上のユーザプロセスの変更指示が済んだならば、ユーザレベルプロセススケジューラ１０はユーザプロセス１２－１を動作させたい時間だけ、即ちプロセス管理表１８の参照で得られたＣＰＵ割当時間だけｕｓｌｅｅｐシステムコールによりスリープすることで、図２２のようにユーザプロセス１２－１を動作させる。次のプロセス識別子Ｂをもつユーザプロセス１２－２についても同様であり、プロセス管理表１８に登録されて動作可能状態にある複数のユーザプロセスについて、この処理を繰り返す。

【００９０】図２３は、図３のライブラリとしてユーザプロセス１２－１，１２－２にリンクされた２つのユーザレベルプロセススケジューラ１０－１，１０－２による動作を示す。プロセス識別子Ａのユーザプロセス１２－１にリンクされたユーザレーザプロセススケジューラ１０－１は、図２２の場合と同様、

ｋｉｌｌシステムコールによる実行、停止、

ｐｒｉｏｃｎｔｌシステムコールによる優先度変更、

ｐｒｉｏｃｎｔｌシステムコールによるクラス変更、

のいずれかを行う。

【００９１】次に、ユーザプロセス１２－１にリンクされたユーザレベルプロセススケジューラ１０－１をプロセス管理表のＣＰＵ使用時間だけｕｓｌｅｅｐシステムコールによりスリープする。これにより、図２３のようにユーザプロセス１２－１を動作させる。次のプロセス識別子Ｂをもつユーザプロセス１２－２についても、同様にして動作させる。

【００９２】図２４は、ＣＰＵの割当対象となるユーザプロセスが複数の場合の動作形態である。図２４にあっては、ユーザレベルプロセススケジューラ１０が２つのユーザプロセス１２－１，１２－２を動作する。まずユーザレベルプロセススケジューラ１０は、

ｋｉｌｌシステムコールによる実行、停止、

ｐｒｉｏｃｎｔｌシステムコールによる優先度変更、

ｐｒｉｏｃｎｔｌシステムコールによるクラス変更、

のいずれかにより、プロセス識別子Ａ，Ｂをもつ２つのユーザプロセス１２－１，１２－２を動作可能状態とする。

【００９３】続いてユーザレベルプロセススケジューラ１０は、ｕｓｌｅｅｐシステムコールによりプロセス管理表から得られたユーザプロセス１２－１，１２－２の合計ＣＰＵ時間で決まる一定時間スリープすることで、図示のように２つのユーザプロセス１２－１，１２－２を動作させる。この場合、ユーザレベルプロセススケジューラ１０のスリープ期間にユーザプロセス１２－１，１２－２がどのように動作するかは、それぞれのユーザプロセス１２－１，１２－２の状態によって決まる。例えば、ユーザプロセス１２－１がＣＰＵを手放さなければユーザプロセス１２－２は動作しないし、ユーザプロセス１２－１が直ちにＣＰＵを手放せばユーザプロセス１２－２が動作するようになる。ここでユーザプロセス１２－１，１２－２が実行を停止する原因としては、Ｉ／Ｏ待ち、ｓｌｅｅｐシステムコールの発行等がある。

【００９４】図２５は、図２４の複数のユーザプロセスを複数のＣＰＵからなるマルチプロセッサ上で動作させる場合の形態である。この場合には、図２５（Ａ）（Ｂ）のように、ユーザプロセス１２－１，１２－２のそれぞれにＣＰＵ４８－１，４８－２を割り当てることができるため、異なるＣＰＵ４８－１，４８－２上で同時に２つのユーザプロセス１２－１，１２－２を動作させることができる。図２６は、予め定めた一定時間Ｔで複数のユーザプロセスの動作を保証する実施形態である。ユーザレベルプロセススケジューラ１０は、

ｋｉｌｌシステムコールによる実行と停止、

ｐｒｉｏｃｎｔｌシステムコールによる優先度の変更、

ｐｒｉｏｃｎｔｌシステムコールによるクラス変更、

のいずれかにより、プロセス識別子Ａのユーザプロセス１２－１を一定時間動作させた後、同様にしてプロセス識別子Ｂのユーザプロセス１２－２を一定時間動作させる。

【００９５】図２７は、複数のユーザプロセスを動作可能状態として一定周期で繰り返し動作させる形態の説明図である。この実施形態にあっては、２つのユーザプロセス１２－１，１２－２をスケジューリング対象としており、一定の周期Ｔについてユーザレベルプロセススケジューラ１０は、まずプロセス識別子Ａのユーザプロセス１２－１を、

ｋｉｌｌシステムコールによる実行、停止、

ｐｒｉｏｃｎｔｌシステムコールによる優先度変更、

ｐｒｉｏｃｎｔｌシステムコールによるクラス変更、

のいずれかで動作可能状態とした後、一定時間動作させ、次に、同様にしてユーザプロセス１２－２を動作さ

せ、これを周期的に繰り返す。この図２５の周期Ｔによる複数のユーザプロセスの動作の繰返しは、各ユーザプロセスの動作周期が同じ場合に適用できる。

【００９６】図２８は、複数のユーザプロセスの動作周期が異なる場合の動作形態である。まずユーザプロセス１２－１は、図２８（Ａ）のように、周期Ｔで動作を繰り返している。これに対しユーザプロセス１２－２は、図２８（Ｂ）のように、２倍の周期にＴで動作を繰り返している。このように２つのユーザプロセス１２－１，１２－２の周期が異なる場合には、図２９のように、複数の動作周期の最小公倍数である周期２Ｔをスケジューリング周期とする。

【００９７】即ち、周期Ｔのユーザプロセス１２－１を動作した後、周期２Ｔのユーザプロセス１２－２を動作し、更に周期Ｔのユーザプロセス１２－１を動作させ、残りを空き周期とし、これを繰り返す。ここで一定周期Ｔで複数のユーザプロセスを動作させる図２７の実施形態にあっては、１周期Ｔ内のＣＰＵ時間を１００％使用しており、このような動作環境に周期Ｔ内で例えば２０％のＣＰＵ時間を要求する別のユーザプロセスをスケジュール対象に加入しようとしても、この要求はユーザレベルプロセススケジューラ１０により拒否されてしまう。

【００９８】これに対し、もしユーザプロセス１２－１，１２－２の周期Ｔ内でのＣＰＵ使用率の合計が８０％以下に制限されていれば、他のユーザプロセスのＣＰＵ時間２０％の要求を受理してユーザレベルプロセススケジューラのスケジュール対象に加入させることができる。そこで、図２７の一定周期Ｔの動作形態についてクオリティ・オブ・セレクト制御の１つであるアドミッションコントロールを採用し、１周期Ｔ内におけるユーザプロセス１２－１，１２－２のＣＰＵ使用時間を８０％以下に制限する。これによって、新たな２０％以内のＣＰＵ使用率をもった他のユーザプロセスの加入要求を効率良く受け入れることができる。

【００９９】図３０は、一定期間内でＣＰＵを分割してユーザプロセスに割り当てる動作形態である。即ち図26または図２７にあっては、一定時間Ｔまたは一定周期内に連続してユーザプロセスのＣＰＵ割当てを行っているが、図３０にあっては、一定期間のＣＰＵ時間を複数に分割し、各分割時間についてユーザプロセス１２－１とユーザプロセス１２－２を割り当てている。

【０１００】図３１は、タイムシェアリングクラスに存在するスケジューリング対象となっているユーザプロセスに対し、強制的に一定のＣＰＵ時間割当てを行う動作形態の説明図である。図３１において、ユーザプロセス１２－１，１２－２はリアルタイムクラス２８に存在し、ユーザプロセス１２－３，１２－４はタイムシェアリングクラス３０に存在している。この場合、ユーザレベルプロセススケジューラ１０は、

ｋｉｌｌシステムコールによる実行停止、

ｐｒｉｏｃｎｔｌシステムコールによる優先度変更、

ｐｒｉｏｃｎｔｌシステムコールによるクラス変更、

のいずれかで、リアルタイムクラス２８にあるユーザプロセス１２－１，１２－２を順次一定時間動作させた後、ユーザプロセス１２－１，１２－２を実行停止状態とし、ｕｓｌｅｅｐシステムコールにより、一定時間ＣＰＵを手放す。

【０１０１】これによりＣＰＵはタイムシェアリングクラス３０に割り当てられ、ユーザプロセス１２－３，１２－４を動作させる。この場合のユーザプロセス１２－３，１２－４のスケジューリングは、オペレーティングシステムＳｏｌａｒｉｓ２．４がもつタイムシェアリングスケジューラが行うことになる。図３２は、本発明のユーザレベルプロセススケジューラのスケジュール空間となるリアルタイムクラスとグローバルな優先度の対応関係を示したマッピングの説明図である。

【０１０２】本発明のユーザレベルプロセススケジューラをサポートするオペレーティングシステムＳｏｌａｒｉｓ２．４では、タイムシェアリングクラス３０については、図３２（Ｃ）のように、タイムシェアリングクラス優先度０～５９をグローバル優先度０～５９にマッピングさせ、設定しており、リアルタイムクラス２８については、図３２（Ａ）のように、リアルタイムクラス優先度０～５９をグローバル優先度１００～１５９にマッピングさせ、設定している。

【０１０３】この図３２（Ａ）の優先度のマッピングは、オペレーティングシステムが起動するときにリンクするＲＴ－ＤＰＴＢＬというファイルに転記されている。本発明にあっては、図３２（Ａ）のマッピングファイルを、図３２（Ｂ）に示すようにリアルタイムクラス２８の最低優先度をタイムシェアリングクラス３０の最低優先度０と等しくする。

【０１０４】図３２（Ｂ）のマッピングによれば、図３３の動作形態が得られる。図７のプロセス優先度制御部２６に対する指示でユーザプロセス１２－２を優先度０に設定することで、一定時間をタイムシェアリングプロセス用時間Ｔｔｓに割り当てる。この場合、タイムシェアリングプロセス用時間Ｔｔｓの間、実際には実行可能なタイムシェアリングプロセスが存在しなかったとすると、図３２（Ｂ）の最低優先度０をもつリアルタイムクラス２８のユーザプロセス１２－２をタイムシェアリングクラスのユーザプロセスの代わりに動作させることができる。

【０１０５】もちろん、このときユーザプロセス１２－１は実行停止状態となっていることから、ユーザプロセス１２－２の動作中に動作することはない。これによって、一定時間タイムシェアリングクラスのユーザプロセスに動作時間を割り当てても効率良くリアルタイムクラスのユーザプロセスを動作させることができる。

５．ユーザプロセスのブロック検出

図３４は、ユーザレベルプロセススケジューラ１０によりユーザプロセス１２－１に時間Ｔ１からＴ４までＣＰＵ時間Ｔを割り当てた状態でブロックが発生した場合である。即ち、時間Ｔ１でユーザプロセス１２－１がＣＰＵ時間の割当てを受けて動作を開始したが、時刻Ｔ２でＩ／Ｏ待ち等でブロック５４が発生したとする。

【０１０６】このブロック５４は適宜のブロック検出手段により検出されて、ブロック検出通知５６がユーザレベルプロセススケジューラ１０に通知される。ユーザレベルプロセススケジューラ１０はブロック検出５６の通知を受けると、残り時間Ｔ３～Ｔ４をタイムシェアリングクラスのユーザプロセスに割り当てる。図３５は、図３４におけるブロック検出のためのプロセスの動作形態である。

【０１０７】図３５において、優先度１５９のユーザレベルプロセススケジューラ１０のスケジュール対象となるユーザプロセス１２－１と同じ優先度１５８にブロック検出プロセス６０を設けている。本発明をサポートするオペレーティングシステムＳｏｌａｒｉｓ２．４のリアルタイムクラスでは、同一優先度に複数の実行可能なプロセスが存在する場合、ＣＰＵ割当ての最小単位であるタイムカンタムごとにラウンドロビンで実行可能プロセスを切り替える。またタイムカンタムは、プロセスごとにｐｒｉｏｃｎｔｌシステムコールで設定することができる。

【０１０８】図３５にあっては、ユーザレベルプロセススケジューラ１０がユーザプロセス１２－１とブロック検出プロセス６０を次の手順で設定し、図３４のようにユーザプロセス１２－１がブロックしたことをブロック検出プロセス６０で検出し、オペレーティングシステムのシグナル機能を使用してユーザレベルプロセススケジューラ１０に通知することになる。

【０１０９】即ち、図３５のユーザレベルプロセススケジューラ１０によるブロック検出プロセス６０の設定は次のようになる。

ユーザプロセス１２－１を優先度１５８に設定する。

ユーザプロセス１２－１のタイムカンタムを、ユーザプロセス１２－１を与えるＣＰＵ時間と等しい時間またはそれ以上の値をもつ時間に設定する。

【０１１０】　ユーザプロセス１２－１を実行可能状態とする。

ブロック検出プロセス６０を優先度１５８に設定する。

ブロック検出プロセス６０のタイムカンタムを適当な時間に設定する。

ブロック検出プロセス６０を実行可能状態とする。

ユーザプロセス１２－１に与えたＣＰＵ時間だけスリープする。

【０１１１】図３７はブロック検出プロセスの他の動作形態であり、ユーザレベルプロセススケジューラ１０のスケジュール対象となるユーザプロセス１２－１に対し更に下位の優先度１５８にブロック検出プロセス６０を設定している。即ち、ユーザプロセス１２－１がブロックしたことをブロック検出プロセス６０で検出し、オペレーティングシステムのシグナル機能を使用してユーザレベルプロセススケジューラ１０に通知する。この場合のユーザプロセス１２－１及びブロック検出プロセス６０の設定は次のようになる。

【０１１２】　ユーザプロセス１２－１を優先度１５８に設定する。

ユーザプロセス１２－１を実行可能状態とする。

ブロック検出プロセス６０を優先度１５７に設定する。

ブロック検出プロセス６０を実行可能状態とする。

ユーザプロセス１２－１を動作させたい時間スリープする。

【０１１３】この図３７のブロック検出プロセス６０の設定の場合の動作も、図３６と同じになる。図３８は、ユーザプロセスに割り当てた一定のＣＰＵ時間内にブロック検出が行われた後、実行可能状態となった場合の動作形態の説明図である。まずユーザレベルプロセススケジューラ１０は、ユーザプロセス１２－１に時間Ｔ１からＴ６までの時間をＣＰＵ時間として割り当てている。しかしながら、ユーザプロセス１２－１が時間Ｔ２でＩ／Ｏ待ち等でブロックし、ブロック検出６８を受けてユーザレベルプロセススケジューラ１０は残り時間Ｔ３からＴ６をタイムシェアリングクラスのプロセス１２０に割り当てる。

【０１１４】ところが、その途中の時間Ｔ４でユーザプロセス１２－１が実行可能になったとすると、適宜の検出手段により実行可能検出７２がユーザレベルプロセススケジューラ１０に通知され、この通知を受けて時間Ｔ５からＴ６を再度ユーザプロセス１２－１に割り当てる。図３８における一度ブロックしたユーザプロセス１２－１の実行可能状態の検出は、オペレーティングシステムからの通知により可能となる。

【０１１５】図３９は、ユーザプロセスの生成と終了を検出した場合の動作形態の説明図である。まず図３９（Ａ）は、ユーザプロセス１２－１のみを動作しており、ＣＰＵ時間の一部をタイムシェアリングプロセス１２０用に割り当てていた場合である。この状態で新たなユーザプロセス１２－２が生成されると、図３９（Ｂ）のように、ユーザプロセス１２－２の生成を検出し、タイムシェアリングプロセスに割り当てていたＣＰＵ時間をユーザプロセス１２－２に割り当てる。

【０１１６】一方、ユーザレベルプロセススケジューラ１０は、図３９（Ｂ）のように、ユーザプロセス１２－１，１２－２にＣＰＵ時間を割り当てていたが、ユーザプロセス１２－２が終了したことを検出した場合、この

ユーザプロセス１２－２のＣＰＵ割当時間をタイムシェアリングプロセス１２０に割り当て、図３９（Ａ）の動作形態とする。

【０１１７】図３９のようなプロセスの生成、終了については、プロセス生成を行うシステムコールｆｏｒｋとプロセスの終了を行うｅｘｉｔシステムコールで行われるが、Ｃ言語で書かれた図４０（Ａ）のアプリケーション３２の場合、ライブラリ３４を通してオペレーティングシステム１４に通知される。また本発明のユーザレベルプロセススケジューラ１０が存在する場合、図４０（Ｂ）のように、アプリケーションプログラム３２よりプロセス生成のシステムコールｆｏｒｋまたはプロセスの終了のシステムコールｅｘｉｔを発行した場合、ユーザレベルプロセススケジューラ１０に対しメッセージ通信機能をもつ新ライブラリ３４０を設ける。

【０１１８】このため新ライブラリ３４０は、アプリケーションプログラム３２よりシステムコールｆｏｒｋまたはｅｘｉｔを受けると、プロセス間通信機能の１つであるメッセージを使用してユーザレベルプロセススケジューラ１０に生成または終了を通知することができる。この新ライブラリ３４０については、コンパイル時にスタティックリンクされたものは再リンクを行い、実行時にダイナミックリンクされるものに対してはダイナミックリンクライブラリを置き替える。これにより、アプリケーションプログラム３２を書き替える必要はない。

【０１１９】図４０（Ｂ）の形態でユーザプロセスの生成、終了を検出した場合の動作は、図３９（Ａ）（Ｂ）と同じになる。もちろん、ユーザプロセスの生成と終了はオペレーティングシステム１４から直接ユーザレベルプロセススケジューラ１０に通知して行うようにしてもよい。尚、上記の実施例は、固定優先度スケジューリングをリアルタイムクラスとして提供するサンマイクロシステム社のオペレーティングシステムＳｏｌａｒｉｓ２．４を例にとるものであったが、他の固定優先度スケジューリング方式を提供する適宜のオペレーティングシステムについてそのまま適用できる。

【０１２０】また本発明のユーザレベルプロセススケジューラがリンクする固定優先度のスケジューラに対する他のクラスのスケジューラとして、タイムシェアリングスケジューラを例にとっているが、これ以外の優先度可変のスケジューラであってもよいことはもちろんである。

【０１２１】

【発明の効果】以上説明してきたように本発明によれば、マルチメディア処理で必要とされる一定割合のＣＰＵ利用を保証するスケジューラをユーザのニーズに合わせて様々な形態で商用のオペレーティングシステム上に実装することができる。またユーザレベルで実現されているにもかかわらず、振る舞いの信用できないプロセスをスケジューリング対象とすることができるなど、オペレーティングシステム内でスケジューラを実現すると同等のスケジューラをユーザレベルで実装することができる。

【図面の簡単な説明】

【図１】本発明の原理説明図

【図２】ユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図３】並列型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図４】階層型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図５】図３を対象とした階層型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図６】同一優先度に２つのユーザレベルプロセススケジューラを設けた動作環境の説明図

【図７】図２のユーザプロセススケジューラの詳細説明図

【図８】図７のプロセス管理表の説明図

【図９】ユーザプロセスの実行、停止による動作説明図

【図１０】図７のプロセス実行制御部による実行と停止の処理のフローチャート

【図１１】ユーザプロセスの優先度変更による動作説明図

【図１２】図７のプロセス優先度制御部による実行と停止の処理のフローチャート

【図１３】ユーザプロセスのクラス変更による動作説明図

【図１４】図７のクラス変更制御部による実行と停止の処理のフローチャート

【図１５】図７の実行停止を用いたユーザプロセスの加入脱会処理のフローチャート

【図１６】図７の優先度変更を用いたユーザプロセスの加入脱会処理のフローチャート

【図１７】図７のクラス変更を用いたユーザプロセスの加入脱会処理のフローチャート

【図１８】ライブラリを用いたスケジュール変更要求の説明図

【図１９】ライブラリのスケジュール仕様に基づくスケジュール変更要求の説明図

【図２０】図１９のスケジュール仕様の説明図

【図２１】マルチスレッド構成をもつユーザプロセスの説明図

【図２２】図２の動作環境で一定時間に１つのユーザプロセスのみを動作させる説明図

【図２３】図３の動作環境で一定時間に１つのユーザプロセスのみを動作させる説明図

【図２４】ＣＰＵ割当て対象のユーザプロセスを複数動作させた説明図

【図２５】複数のＣＰＵを割り当てて動作させた場合の説明図

31

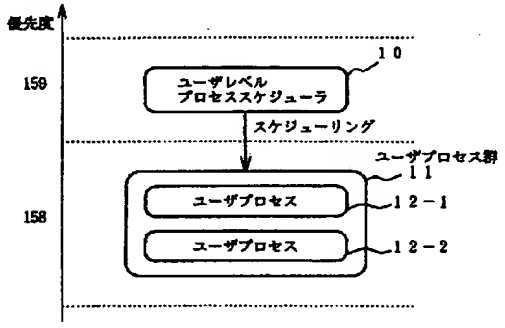
【図２６】一定時間内に複数のユーザプロセスを動作させる説明図
【図２７】一定周期で複数のユーザプロセスを動作させる説明図
【図２８】動作周期の異なるユーザプロセスの説明図
【図２９】図２８のユーザプロセスを一定周期で動作させた説明図
【図３０】一定期間を複数分割して複数のユーザプロセスを動作させた説明図
【図３１】タイムシェアリングプロセスにＣＰＵ時間を割り当てた説明図
【図３２】タイムシェアリングクラスとリアルタイムクラスの優先度のマッピングの説明図
【図３３】リアルタイムクラスの優先度０にマッピングしたユーザプロセスの動作の説明図
【図３４】ユーザプロセスのブロック検出時の動作の説明図
【図３５】ブロック検出プロセスの動作環境の説明図
【図３６】ブロック検出プロセスのブロック検出と動作の説明図
【図３７】ブロック検出プロセスの他の動作環境の説明図
【図３８】ブロック検出後に実行可能が検出された場合の説明図
【図３９】ユーザプロセスの生成と終了の説明図

32

【図４０】アプリケーションプログラムのライブラリを使用したユーザプロセスの発生終了の検出通知の説明図
【符号の説明】
１０，１０－１，１０－２，１０－１１，１０－１２，１－２１，１０－２２：ユーザレベルプロセススケジューラ
１２，１２－１，１２－２～１２－４：ユーザプロセス
１４：オペレーティングシステム（ＯＳ）
１６：クラス変更指示部
１８：プロセス管理表
２０：プロセス実行指示部
２２：クラス変更制御部
２４：プロセス実行制御部
２６：プロセス優先度制御部
２８：リアルタイムクラス（固定優先度スケジュールクラス）
３０：タイムシェアリングクラス（動的優先度スケジュールクラス）
３２：アプリケーションプログラム
３４，３８：ライブラリ
３６：親プロセス
４２：マルチスレッドライブラリ
４４－１～４４－３：スレッド
６０：ブロック検出プロセス
３４０：新ライブラリ

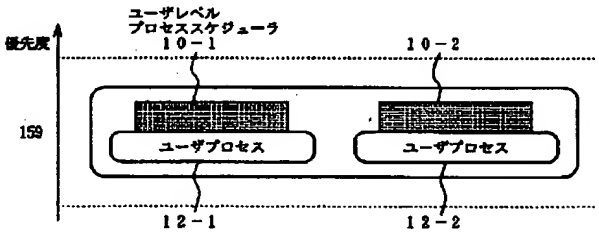
【図２】

ユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



【図３】

並列型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



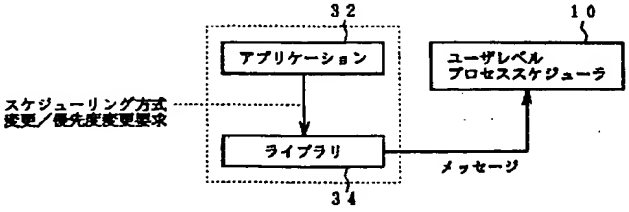
【図８】

図７のプロセス管理表の説明図

プロセス識別子	周期内のＣＰＵ使用時間（ｍｓ）
A	５０
B	２０
C	１０
未使用	－
未使用	－

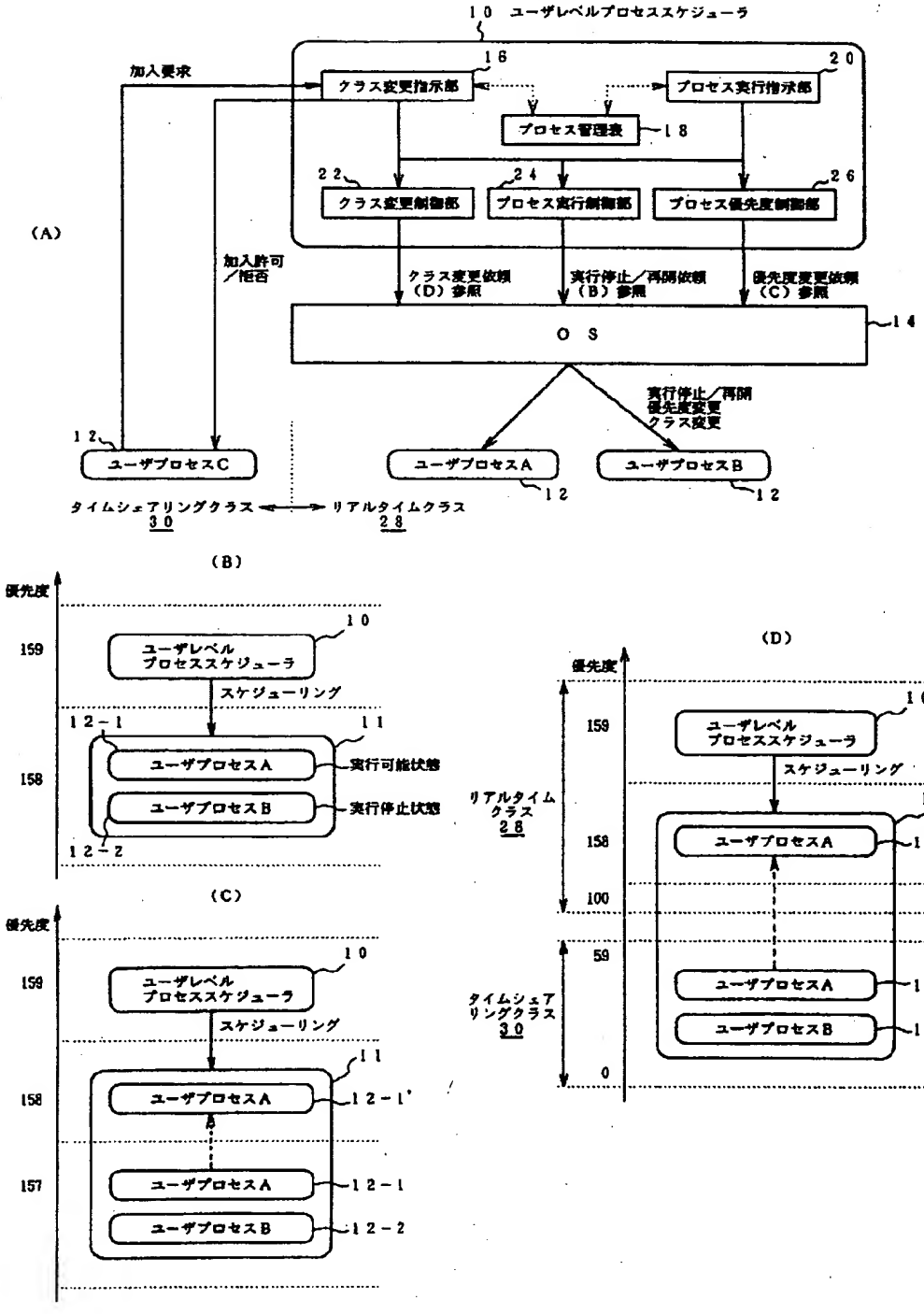
【図１８】

ライブラリを用いたスケジュール変更要求の説明図



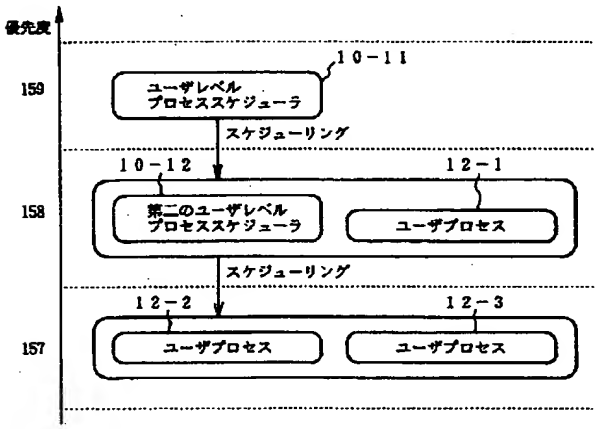
【図１】

本発明の原理説明図



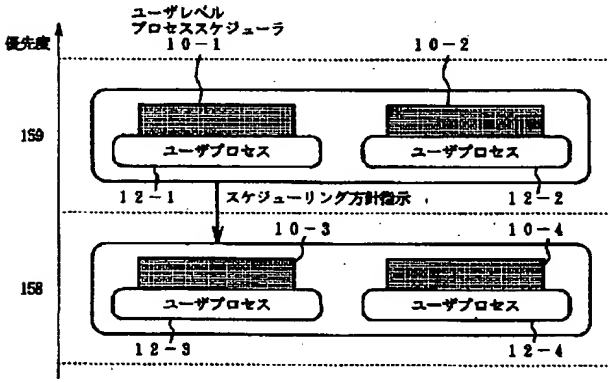
【図４】

階層型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



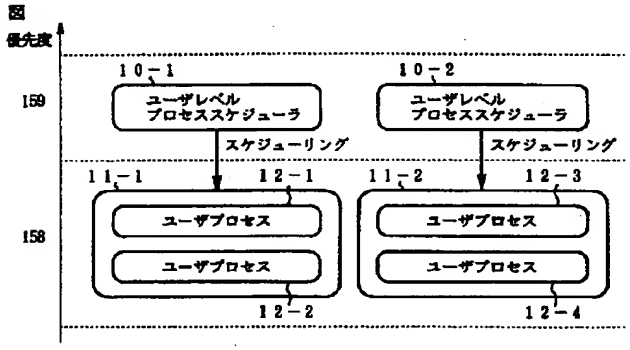
【図５】

図３を対象とした階層型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



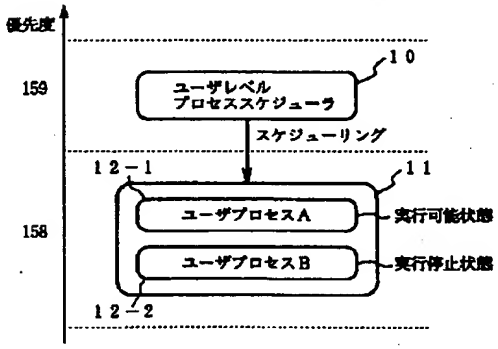
【図６】

同一優先度に２つのユーザレベルプロセススケジューラを設けた動作環境の説明図



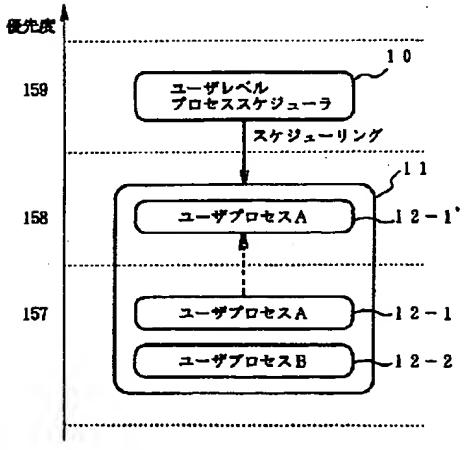
【図９】

ユーザプロセスの実行、停止による動作説明図



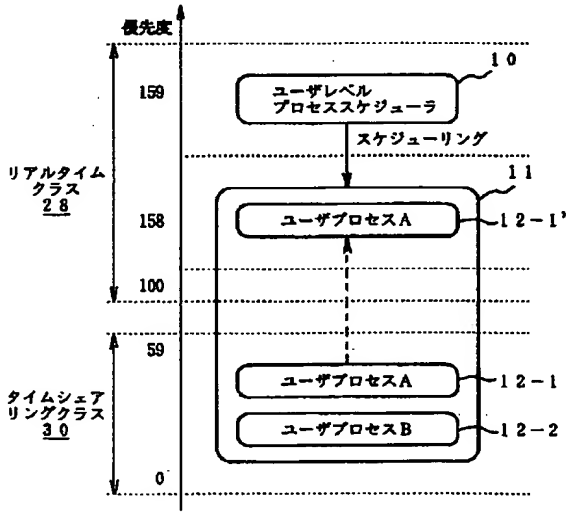
【図１１】

ユーザプロセスの優先度変更による動作説明図



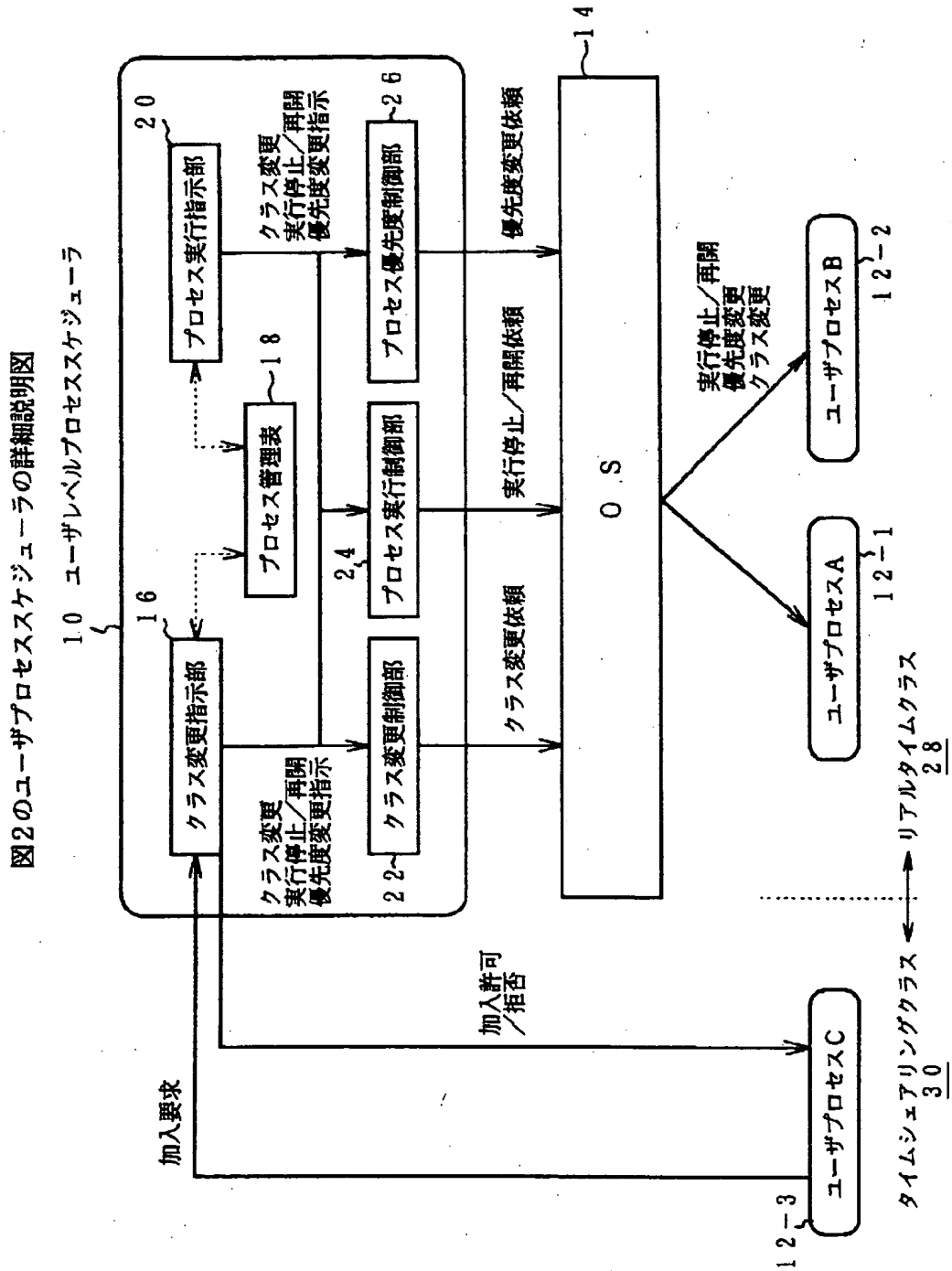
【図１３】

ユーザプロセスのクラス変更による動作説明図



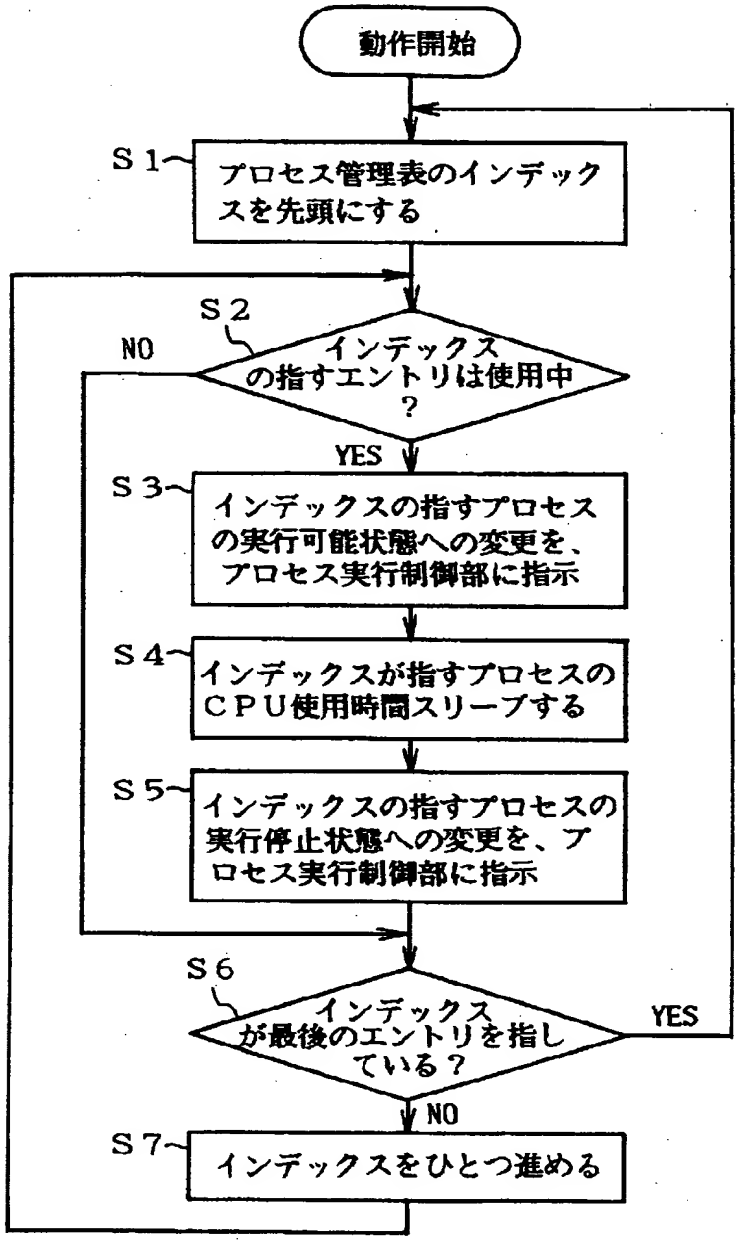
【図７】

図２のユーザプロセススケジューラの詳細説明図



【図１０】

図７のプロセス実行制御部による実行と停止の処理のフローチャート



【図２０】

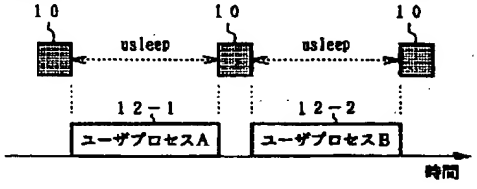
図１９のスケジュール仕様の説明図

40

開始時刻	終了時刻	スケジューリング方式
00:00:00	00:00:30	タイムシェアリングクラス
00:00:30	00:10:00	リアルタイムクラス
00:10:00	00:10:30	タイムシェアリングクラス

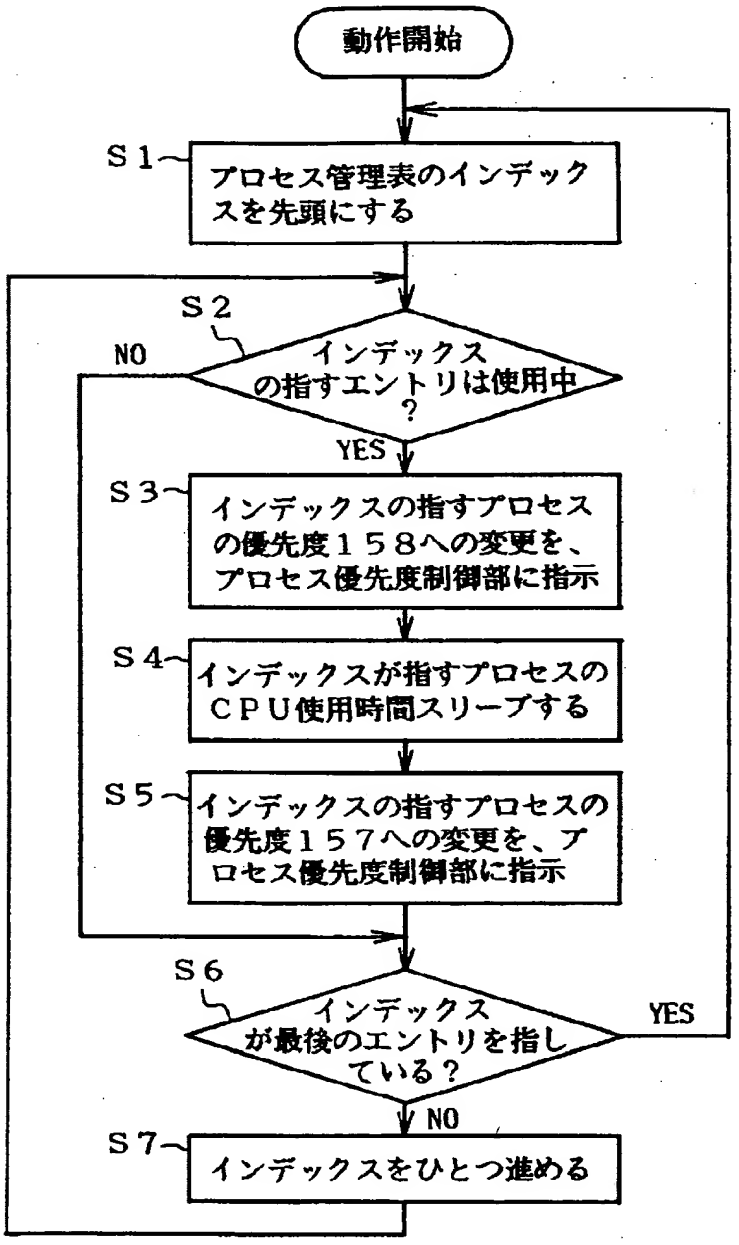
【図２２】

図２の動作環境で一定時間に１つのユーザプロセスのみを動作させる説明図



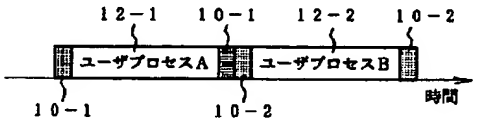
【図１２】

図７のプロセス優先度制御部による実行と停止の処理のフローチャート



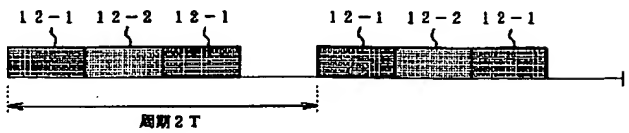
【図２３】

図３の動作環境で一定時間に１つのユーザプロセスのみを動作させる説明図



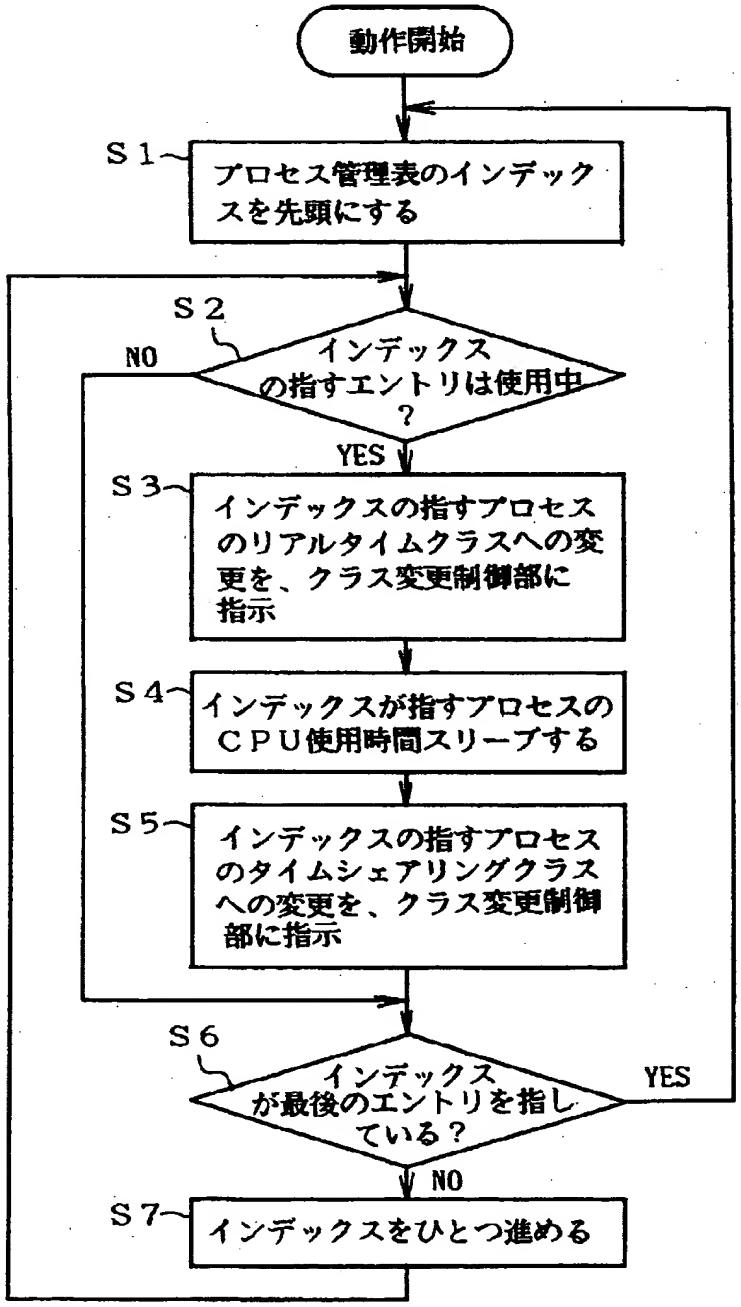
【図２９】

図２８のユーザプロセスを一定周期で動作させた説明図



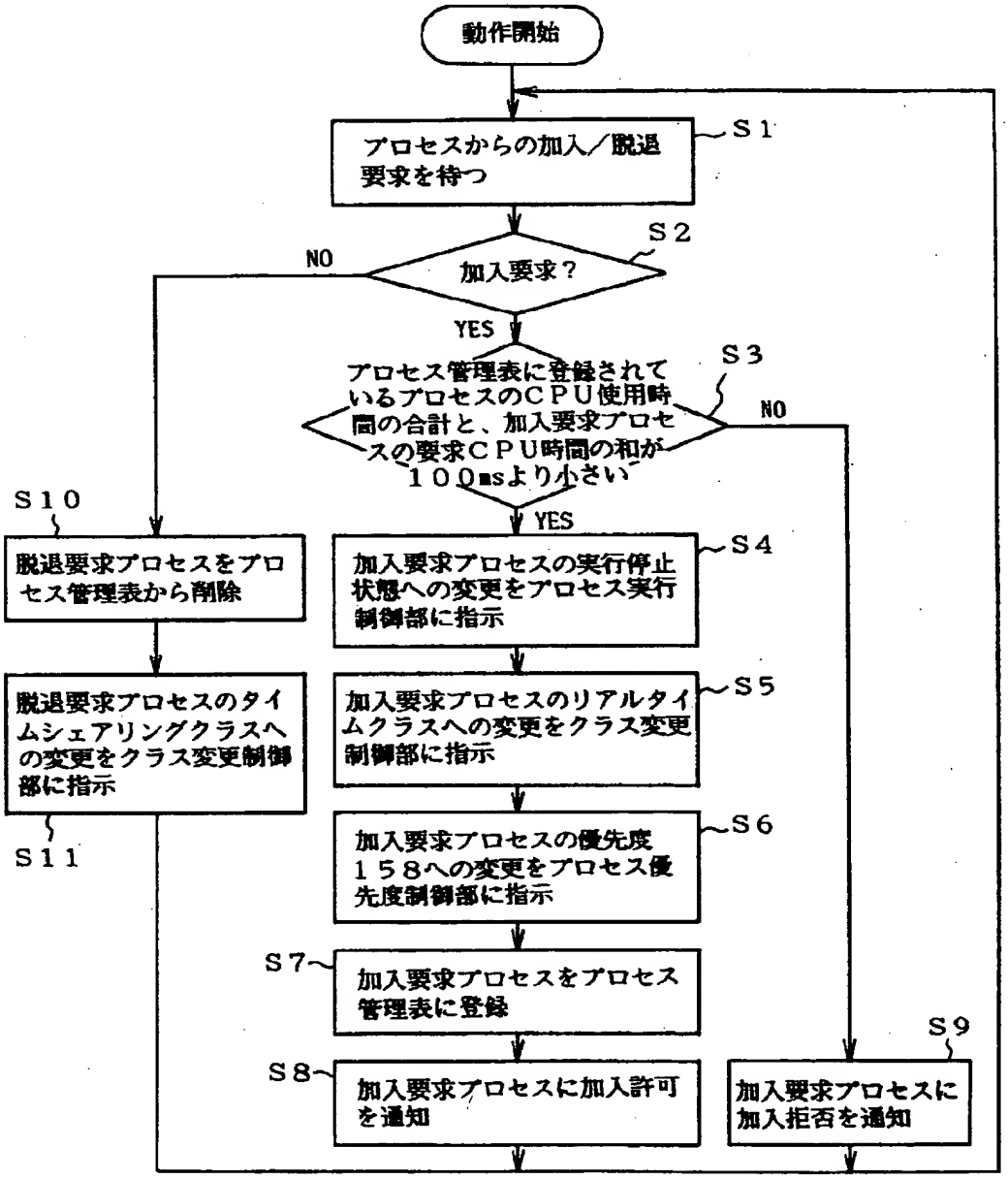
【図１４】

図７のクラス変更制御部による実行と停止の処理のフローチャート



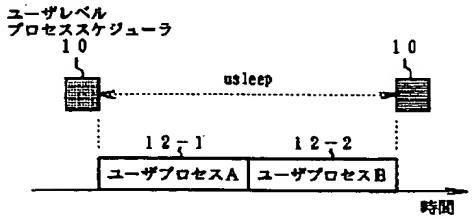
【図１５】

図７の実行停止を用いたユーザプロセスの加入脱会処理のフローチャート



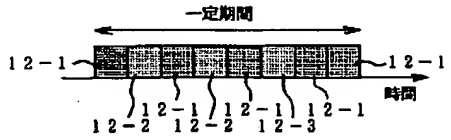
【図２４】

ＣＰＵ割当て対象のユーザプロセスを複数動作させた説明図



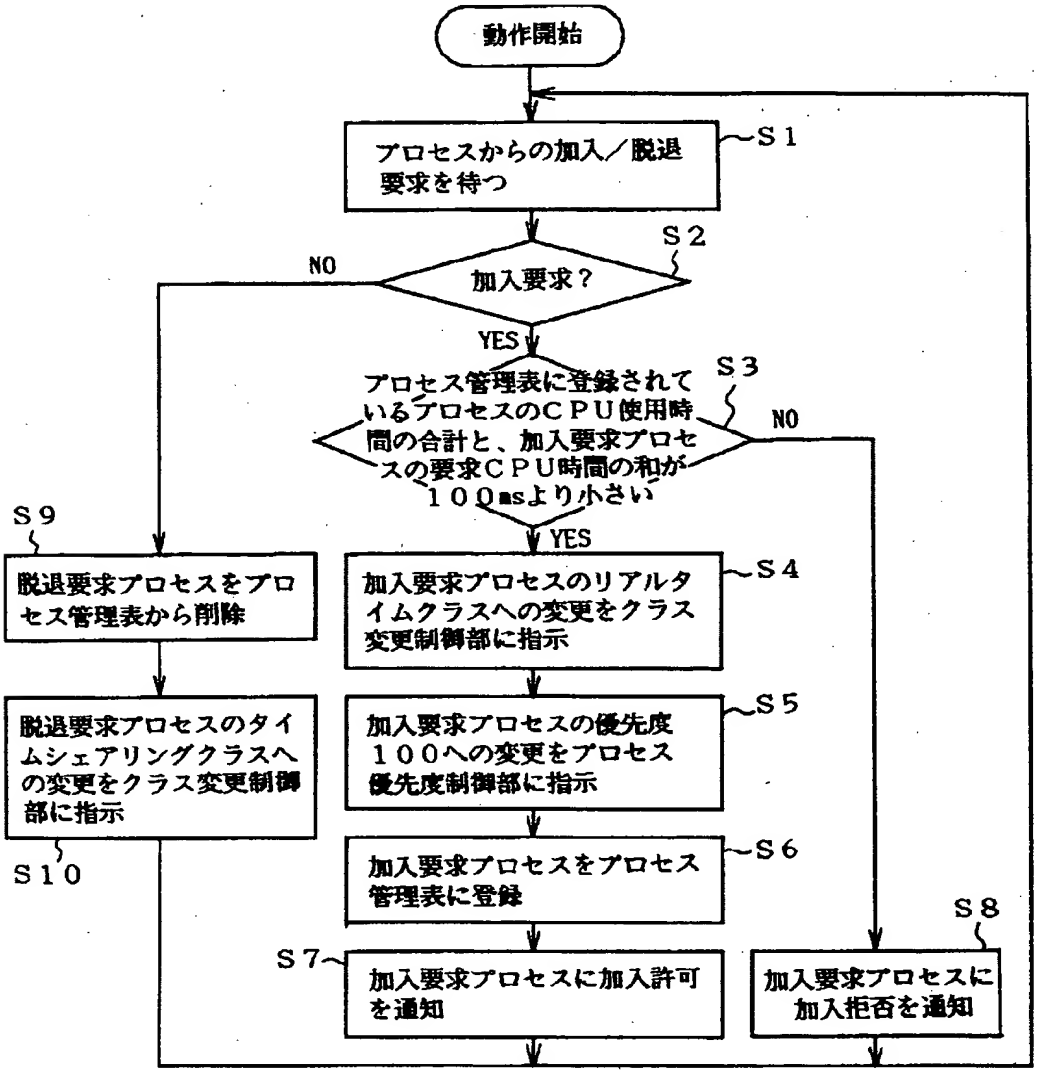
【図３０】

一定期間を複数分割して複数のユーザプロセスを動作させた説明図



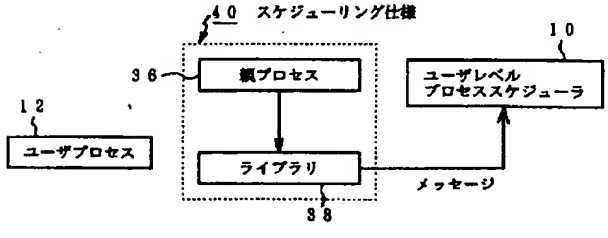
【図１６】

図７の優先度変更を用いたユーザプロセスの加入脱会処理のフローチャート



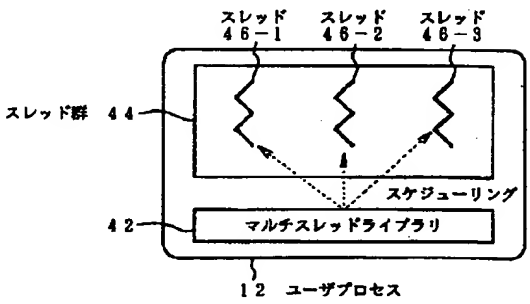
【図１９】

ライブラリのスケジュール仕様に基づくスケジュール変更要求の説明図



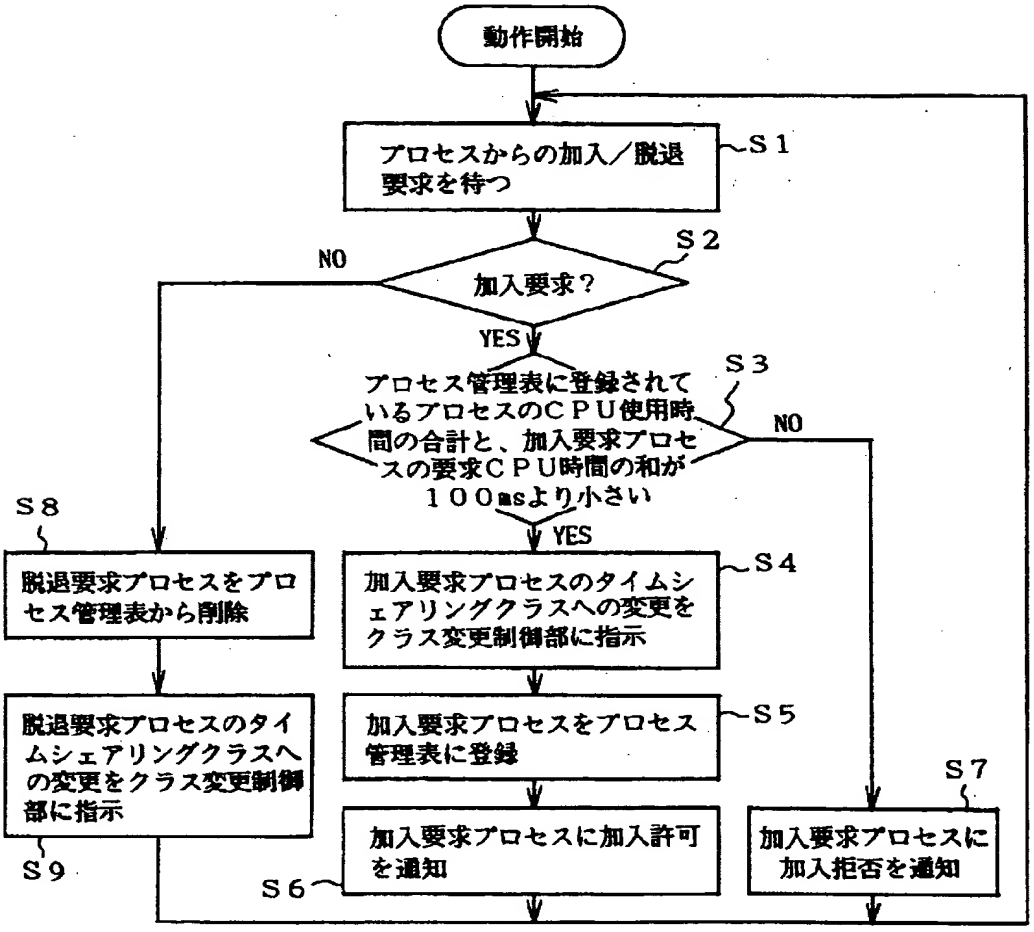
【図２１】

マルチスレッド構成をもつユーザプロセスの説明図

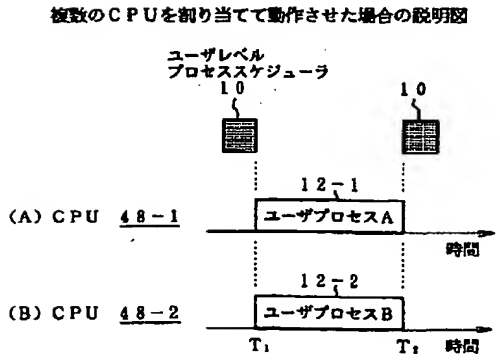


【図１７】

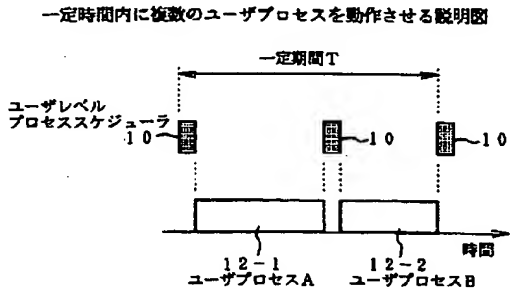
図７のクラス変更を用いたユーザプロセスの加入脱会処理のフローチャート



【図２５】

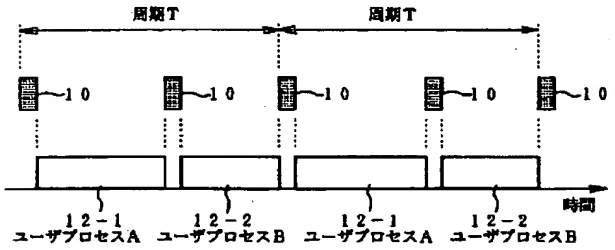


【図２６】



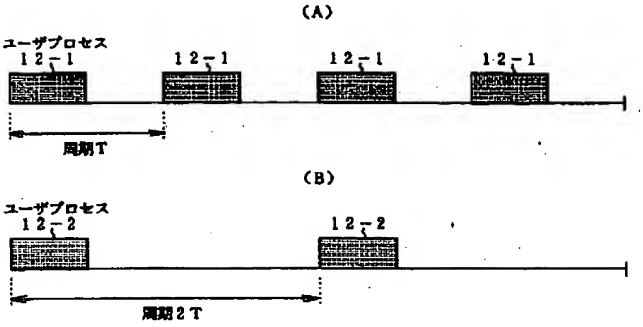
【図２７】

一定周期で複数のユーザプロセスを動作させる説明図



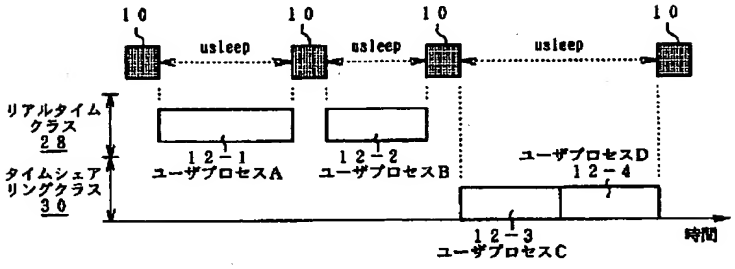
【図２８】

動作周期の異なるユーザプロセスの説明図



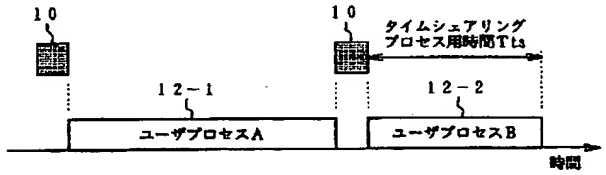
【図３１】

タイムシェアリングプロセスにＣＰＵ時間を割り当てた説明図



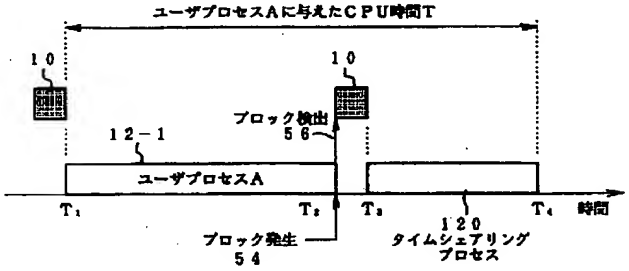
【図３３】

リアルタイムクラスの優先度０にマッピングしたユーザプロセスの動作の説明図



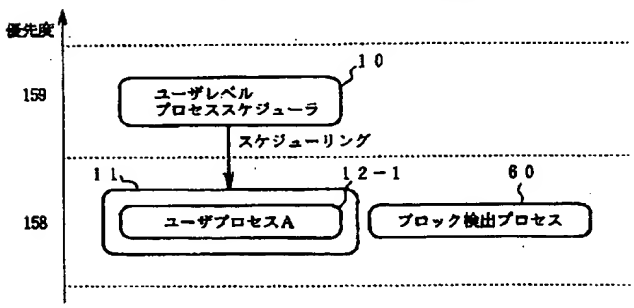
【図３４】

ユーザプロセスのブロック検出時の動作の説明図



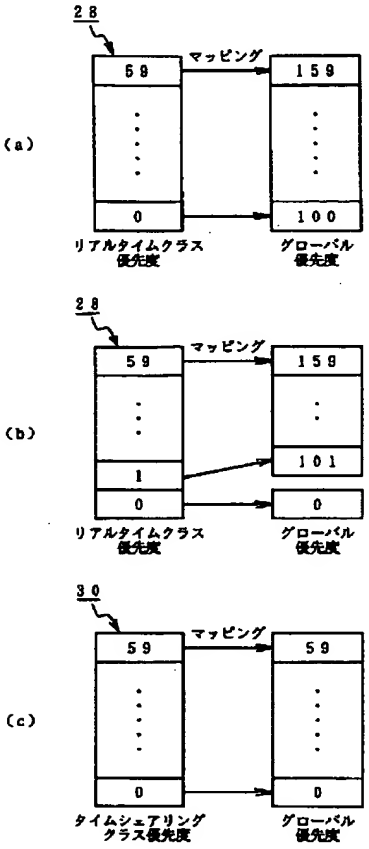
【図３５】

ブロック検出プロセスの動作環境の説明図



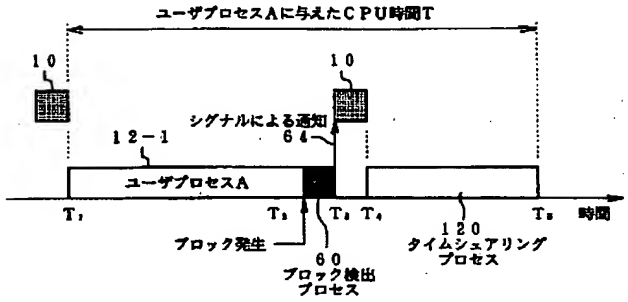
【図３２】

タイムシェアリングクラスとリアルタイムクラスの優先度のマッピングの説明図



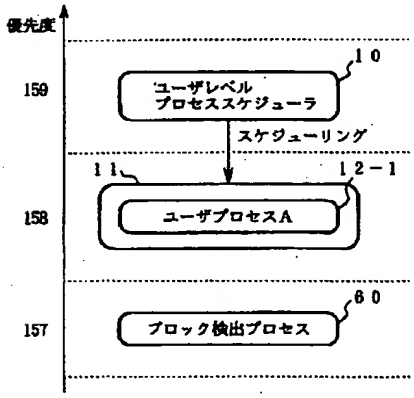
【図３６】

ブロック検出プロセスのブロック検出と動作の説明図



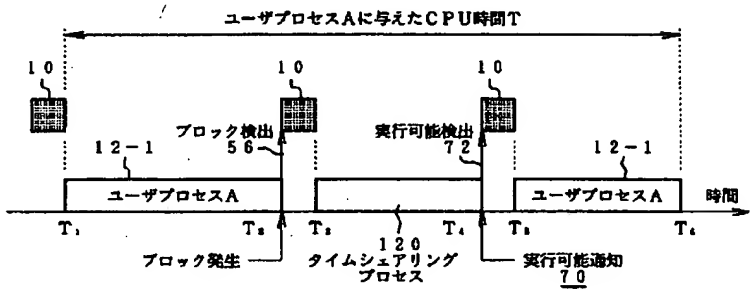
【図３７】

ブロック検出プロセスの他の動作環境の説明図



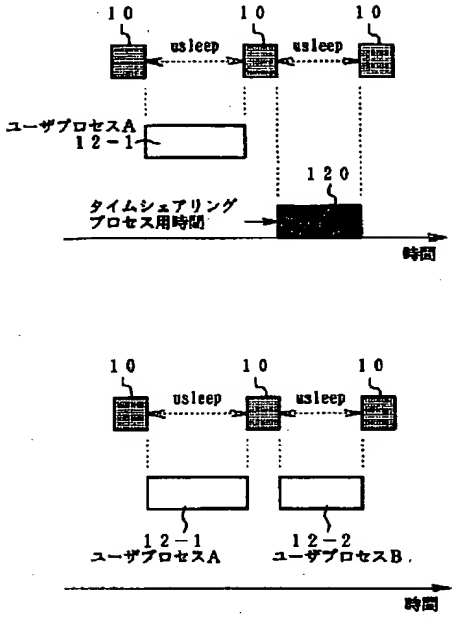
【図３８】

ブロック検出後に実行可能が検出された場合の説明図



【図３９】

ユーザプロセスの生成と終了の説明図



【図４０】

アプリケーションプログラムのライブラリを使用したユーザプロセスの発生終了の検出通知の説明図